Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Trabajo de Fin de Grado

Metodología de aproximación a una percepción cromática análoga entre el medio físico y digital.

Approach method to an analogous chromatic perception between the physical and digital medium.

Autor
Darío Sánchez Salvador

Director
Fernando Palos Mateo

Ponente
Ana Cristina Murillo

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Junio 2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Darío Sánchez Salvador,

con nº de DNI 72999575J en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Metodología de aproximación a una percepción cromática análoga entre el medio físico y digital,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de Junio de 2018

Fdo: Darío Sánchez Salvador

Metodología de aproximación a una percepción cromática análoga entre el medio físico y digital

Resumen

Este proyecto presenta el diseño y desarrollo de un prototipo para la generación de imágenes digitales cuyos colores en pantalla sean percibidos por el ojo humano de forma equivalente a un medio físico como el papel.

En primer lugar, se han analizado las causas que provocan la diferencia entre ambos medios y se ha realizado un estudio a través del cual se ha determinado que la mejor manera de llevar a cabo esta labor de una forma sencilla y eficiente era utilizando la cámara de un dispositivo. Se han analizado los distintos problemas que fueron surgiendo a la hora de capturar imágenes y se han estudiado distintas técnicas planteadas para tratar de contrarrestarlos. De la misma forma, se ha estudiado la posible utilización de distintos dispositivos y a partir de las conclusiones se realizó una selección del hardware a utilizar en el proyecto.

Como parte central, se ha propuesto un algoritmo que, a partir de imágenes capturadas por una cámara, es capaz de hallar colores equivalentes representados en una pantalla. Para ello se han utilizado dos dispositivos: La pantalla a calibrar y el controlador que captura imágenes, las analiza y modifica los valores de la pantalla para intentar ajustarlos a una muestra previamente establecida. Se ha realizado una evaluación exhaustiva de los resultados y las prestaciones de dicho algoritmo tras ejecutar una batería de pruebas de igualación de colores. Para su uso se ha implementado una aplicación en Android que contenga este algoritmo y permita el uso de la cámara del dispositivo.

Como forma de verificación y aplicación a un problema concreto, se ha planteado la generación de láminas de test de Ishihara. Este test está formado por una colección de láminas donde cada una de ellas contiene una distribución de puntos coloreados. En esta distribución, los colores que pertenecen a la misma línea de confusión forman figuras únicamente perceptibles por personas sin ninguna deficiencia en la percepción del color, mientras que las personas con anomalías perciben una forma distinta o ninguna. Una lámina está formada por un conjunto de alrededor de diez colores. Para la generación de estos test de forma digital, debe calcularse un equivalente digital para cada uno de los colores de una lámina, que posteriormente sustituyan a los de una versión escaneada de la misma.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación: La digitalización en la optometría

En los últimos años el mundo de la optometría ha experimentado un cierto empuje hacia la digitalización de sus servicios, un movimiento que no siempre se antoja fácil. Aún en pleno 2018 la gran mayoría de los instrumentos de medición y corrección de problemas visuales continúan siendo analógicos. Pese a ello, algunas empresas continúan desarrollando herramientas digitales con la esperanza de lograr una forma más asequible e igualmente efectiva de diagnosticar, medir, prevenir y tratar deficiencias visuales. Este es el caso de SmarThings4Vision, una SpinOff que cuenta con la participación de la Universidad de Zaragoza y con la que se ha desarrollado el presente Trabajo de Fin de Grado en concepto de colaboración becada.

Figura 1.1: Sistema OptoTab de test de detección de deficiencias visuales.

SmarThings4Vision se especializa en el desarrollo de aplicaciones sobre el sistema operativo Android para la prevención, diagnóstico y tratamiento de deficiencias visuales mediante medios digitales. Su producto estrella es OptoTab con sus variantes Office Polar, Office y Screening, centrado en ofrecer una experiencia más cómoda tanto para optometristas como para pacientes a la hora de

realizar diversos test y ejercicios de visión. Estos productos se componen de una pantalla (polarizada o no dependiendo de la versión) y un tablet desde donde se eligen y controlan los test y sus parámetros (Fig. 1.1).

El presente proyecto ha sido financiado por SmarThings4Vision tanto para ampliar su conocimiento en un nuevo área de interés como para una posible futura implementación de sus conclusiones para sus productos, permitiendo añadir test de detección de anomalías en la percepción cromática a su ya amplio catálogo.

En las aplicaciones de optometría digitalizadas es esencial una representación correcta del color en los medios digitales. Para ello hay que tratar el problema de la diferente forma en la que se representa el color en un medio digital como una pantalla frente a un equivalente físico como sería, por ejemplo, el papel. Estas diferencias resultan, a menudo, en dificultades para la reproducción del color en entornos donde el tratamiento de éste es especialmente delicado, siendo uno de ellos el campo de la optometría. Por lo tanto, como se detalla a continuación, el objetivo de este trabajo es investigar este problema y proponer una posible solución.

1.2. Objetivos

El objetivo general de este proyecto es estudiar el problema de la representación del color en medios digitales cuando se requiere cierta fidelidad respecto al medio físico y plantear un prototipo que reproduzca estos colores con una precisión aceptable para un objetivo determinado. Para ello, es necesario:

- Realizar un estudio sobre el funcionamiento y características de la cámara de los dispositivos disponibles para el proyecto, con el objetivo de valorar su aplicabilidad en una solución al problema propuesto.
- Implementar un algoritmo de "color matching"que al tomar una muestra a través de la cámara, sea capaz de calcular un color de forma automática que, al ser representado por la pantalla objetivo, sea percibido por el usuario de forma análoga.
- Diseñar e implementar un prototipo para el sistema operativo Android que haga uso de la cámara y del algoritmo mencionado anteriormente para que, de forma automática, calcule un color representable equivalente a otro seleccionado.
- Como objetivo adicional se busca aplicar esta metodología a la representación de test para diagnóstico de anomalías cromáticas en la visión. Para ello se realizará, con carácter académico, una representación del test de Ishihara en el cual se sustituirá su paleta de color por la equivalente obtenida en el proceso.

1.3. Trabajo previo

La problemática descrita anteriormente, en la que la representación del color en medios digitales no es igual que en medios físicos, tiene su origen en el principio físico del color. Por lo tanto, para entender este trabajo, es importante

resumir este principio, estudiar las técnicas y medios existentes para calibrar el color en medios digitales y finalmente exponer de forma breve el funcionamiento de los test de detección de deficiencias cromáticas.

Principio físico del color. El color en los objetos se basa en la existencia de una fuente de luz externa cuyos rayos, al incidir sobre las distintas superficies, son parcialmente absorbidos dependiendo de su composición química, siendo el color resultante la combinación de los rayos no absorbidos (Figura 1.2). En este caso, al añadir colores se añaden pigmentos que absorben rayos de distintos colores que se sustraen del blanco recibido, por lo tanto, es la combinación de los rayos que no absorben dichos pigmentos los que componen el color. A este fenómeno se le denomina como *mezcla sustractiva del color*, representado en a Figura 1.4 a la derecha. La suma de todos los colores en una mezcla sustractiva forma el color negro, al absorber todos los rayos.[1]

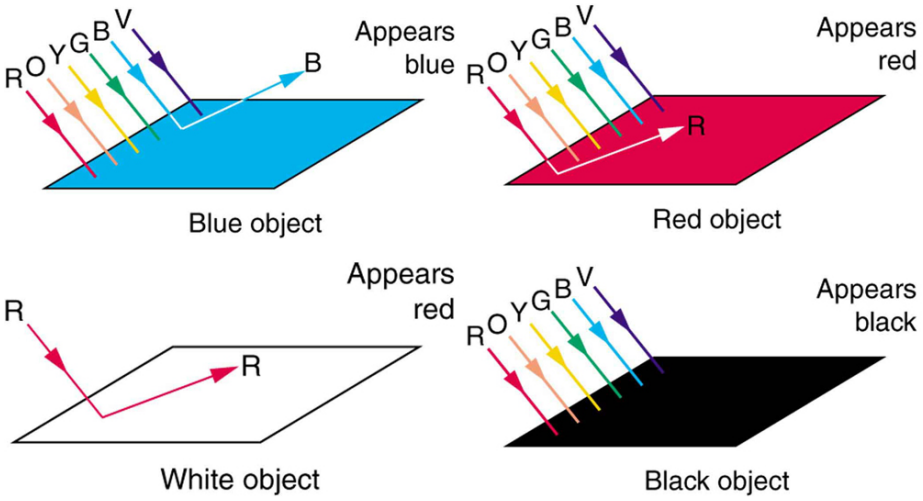

Figura 1.2: Esquema de la física básica del color en objetos. Fuente: OpenStaxPhysics[1]

Por otro lado, el color en el medio digital funciona de una manera diferente. En este caso no se depende de una fuente de luz externa, sino que es la propia generación de luz la que forma el color. En las pantallas el color se genera como la mezcla de rayos de luz de tres colores primitivos: Rojo, Verde y Azul (R, G y B basado en sus iniciales en inglés, respectivamente). Dado que la fuente de luz es la misma que el origen del color, los rayos ahora se mezclan de forma inversa: Más colores significa más rayos agrupados. A este fenómeno se le conoce como *mezcla aditiva del color*, representado en la Figura 1.4 a la izquierda. Esto significa que cuando se emiten todos los colores, se generan todos los rayos, lo cual da lugar al color blanco. En la Figura 1.3 se representa el funcionamiento de esta mezcla.

Estos tres colores primarios suelen estar contenidos en una única unidad física llamada **píxel**. A modo de ejemplo, cualquiera de los tríos de la Figura 1.3 forma un píxel. Las pantallas crean las formas y colores mediante cuadrículas de

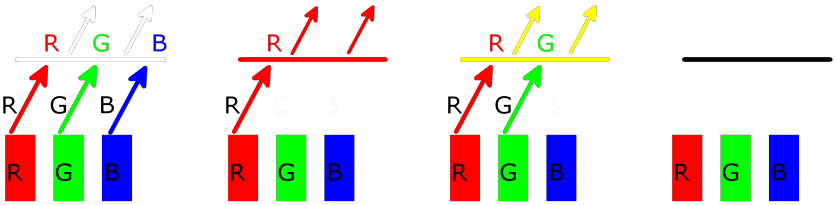

Figura 1.3: Esquema de la física del color en pantallas.

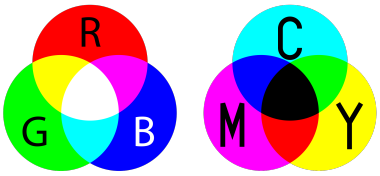

Figura 1.4: Mezcla de colores aditiva (RGB, izq) y sustractiva (CMY, der). Fuente: Wikimedia, Mike Horvath, 2018

píxeles, donde cada píxel es capaz de representar un conjunto limitado de colores ajustando la intensidad de sus primarios. Las pantallas modernas contienen gran cantidad de píxeles y el ordenamiento de estas cuadrículas varía dependiendo de la tecnología. En la Figura 1.5 se pueden observar algunas de las más comunes.

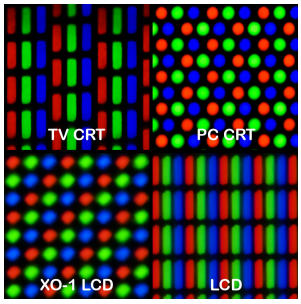

Figura 1.5: Ejemplos de populares geometrías de píxeles. Fuente: Wikimedia, Peter Halasz, 2009

Espacios de color. Los espacios de color representan el conjunto de colores representables en tuplas de números (tres en caso del RGB). Todos los espacios de color necesitan estar basados en un espacio absoluto para ser entendidos, si no, son simplemente una forma arbitraria de definir colores. Los espacios absolutos son aquellos en los que:

- La diferencia percibida entre dos colores está directamente relacionada con su distancia en el espacio vectorial representada por dos puntos. [2]
- Sus colores son inequívocos, esto es, están basados en mediciones colorimétricas sin referencias externas. [3]

El más antiguo y usado es el CIEXYZ. Cualquier punto en este espacio está definido por tres coordenadas x, y y z tal que:

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z}$$
$$z = \frac{Z}{X+Y+Z}$$

Donde los valores X, Y y Z son calculados a través de mediciones realizadas por los creadores y disponibles en distintas fuentes.

En la Figura 1.6 puede observarse representado un espacio de color CIE XYZ en un espacio tridimensional en el que se aprecia como los valores cercanos al 0 provocan un oscurecimiento y finalmente la ausencia de color (color negro).

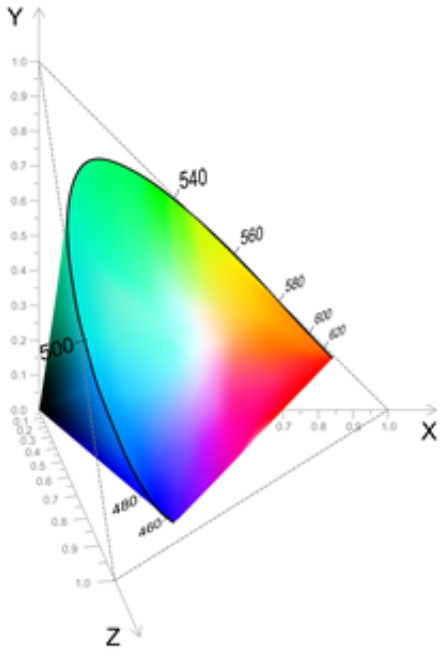

Figura 1.6: Espacio de color CIE XYZ representado en un espacio tridimensional. Fuente: `http://www.habr.com/post/181580/`, 2013

En este espacio se suelen basar los espacios de color digitales, utilizando RGB salvo contadas excepciones. Un subespacio RGB dispone de tres vectores que se utilizan como base para definir su subespacio vectorial, cada uno de ellos correspondiendo a las coordenadas de uno de sus colores primitivos en CIEXYZ. En la Figura 1.7 se observan algunos de los más populares superpuestos sobre un CIE.

Ahora bien, si bien es cierto que los espacios de color utilizados comúnmente en las pantallas son subespacios de dimensión 3 pertenecientes al mismo espacio absoluto CIE, casi nunca sus espacios de color particulares (dependientes

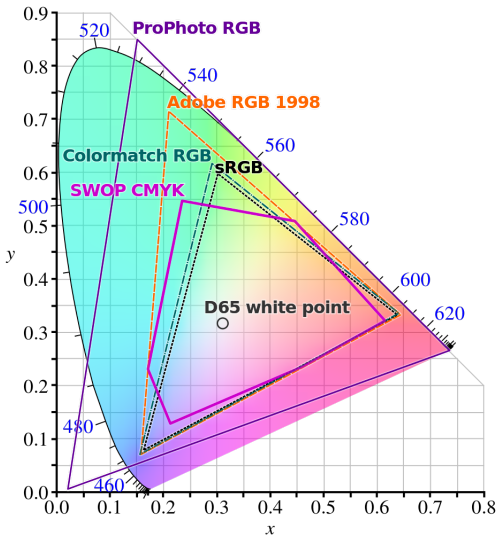

Figura 1.7: Espacios de color populares sobre el CIEXYZ. Fuente: Wikimedia, BenRg and cmglee, 2014

de su implementación tanto hardware como software) coinciden entre distintos dispositivos.

Esta realidad dificulta e incluso puede llegar a imposibilitar prácticas como la realización de diagnósticos de deficiencias en la percepción del color con medios digitales alternativos a las tradicionales láminas. Lo que se pretende con este trabajo es darle la vuelta al problema y encontrar una solución sencilla que permita establecer equivalentes cromáticos de una paleta de colores de forma que, a partir de ella, puedan desarrollarse soluciones para una mejor aproximación al color que se desea representar.

Calibración de color. Este trabajo se enmarca a priori dentro del campo de la calibración de color. El objetivo de la calibración de color es ajustar la respuesta de salida de un dispositivo para que coincida con la de un determinado estándar. Para ello existen gran diversidad de métodos, aunque a menudo se reducen a dos alternativas: La creación de un perfil *ICC* y el establecimiento de una relación no estandarizada entre dos espacios de color.

El ICC o *International Color Consortium* es un consorcio creado en 1993 como resultado de un acuerdo entre varias marcas comerciales con intereses en la estandarización de la representación de color. Este acuerdo dio lugar a la creación de la especificación ICC, que permite la transformación entre espacios de colores mediante la creación de *perfiles ICC*, que especifican el "mapping"necesario entre dichos espacios de color. También es posible crear relaciones entre espacios de color sin definir especificaciones ICC, bien creando otro tipo de matrices de corrección o mediante el ajuste de las propiedades de la pantalla.

Los calibradores de color para pantallas son dispositivos hardware normalmente utilizados para la creación de perfiles ICC que posteriormente pueden ser utilizados para la corrección de color. Aunque existen diversas opciones, incluyendo la de código abierto *ColorHug2*, no son dispositivos baratos, pudiendo

llegar a costar en torno a 500€ la unidad para dispositivos de gama media.

Aunque el objetivo del trabajo es parecido, realmente un calibrador de color no soluciona el problema propuesto, pues se busca lo más cercano a una equivalencia cromática entre un medio físico y otro digital, mientras que la calibración de color únicamente garantiza la igual percepción de colores entre medios digitales.

Test de anomalía cromática. La visión cromática anómala o 'daltonismo', como se conoce comúnmente en honor al científico John Dalton, es una deficiencia visual que afecta a la percepción cromática de un individuo, provocando sobre el que la padece la imposibilidad de distinguir entre distintos rangos de colores. Las anomalías en la visión de los colores se clasifican fundamentalmente en congénitas y adquiridas (hereditarias y reversibles, respectivamente). Este trabajo centra su estudio en las anomalías congénitas, de mayor interés científico-técnico.

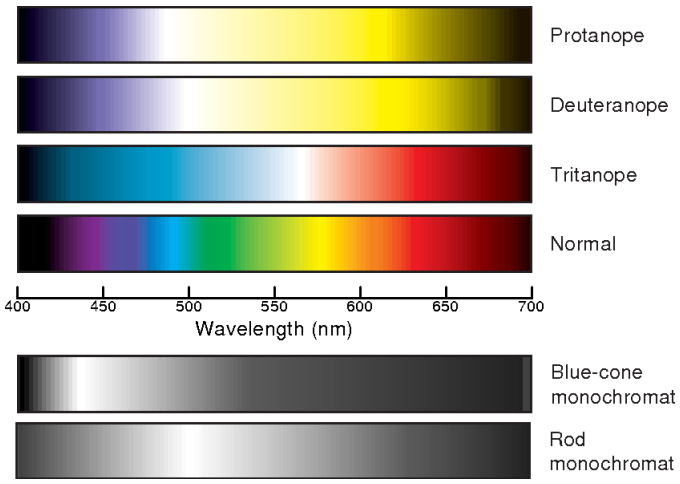

Figura 1.8: Espectro visible de un tricrómata normal y de diferentes anomalías del color.[4]

El origen de las deficiencias cromáticas se basa generalmente en el funcionamiento anómalo (o déficit en algún caso) de los fotorreceptores de la retina que reciben la información luminosa, los **conos**. Existen tres tipos de conos, conos sensibles al rojo, conos sensibles al verde y conos sensibles al azul y, en función de qué tipo de cono y en que magnitud están afectados, podemos distinguir los diferentes tipos de anomalías cromáticas: acromatismo (ausencia de conos), monocromatismo (un solo tipo de cono), dicromatismo (sólo dos conos funcionan correctamente) y tricomatismo anómalo, donde todos los conos son funcionales, pero uno de ellos de forma reducida. Este último tipo se distingue en protán (rojo débil), deután (verde débil) o tritán (azul débil). [5] En la figura 1.9 puede observarse un ejemplo de la diferente percepción de los colores entre una persona normal y una afectada por protanopia.

La figura 1.8 muestra el espectro visible de un tricrómata normal (sin deficiencias cromáticas) y de diferentes anomalías del color.

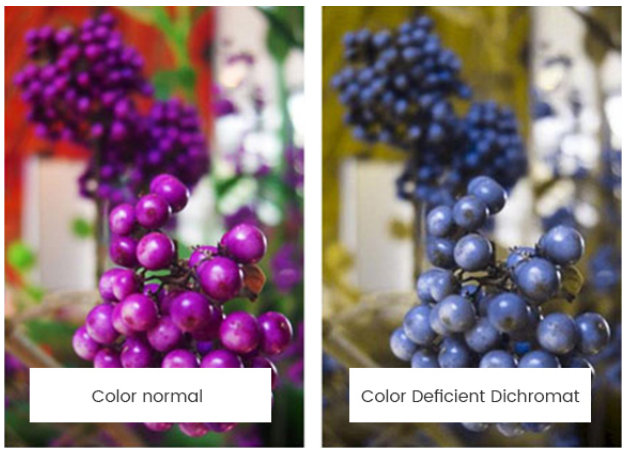

Figura 1.9: Demostración de la diferencia entre la percepción de una persona normal y una afectada por protanopia (discromatopsia de tipo protán). Fuente: `http://www.colorvisiontesting.com/color2.htm`

El test más extendido para el diagnóstico de las anomalías de color es el test de Ishihara (Figura 1.10). Este test, basado en láminas pseudoisocromáticas, fue ideado por primera vez en Japón por el doctor Shinobu Ishihara en 1916 y posteriormente comercializado en todo el mundo en 1917.

Este test consta de 24 láminas y consiste en una distribución de puntos dispuestos de tal modo que representan una figura o línea que puede identificarse o seguirse. Los puntos que conforman los números son visibles por las personas sin alteraciones en el eje rojo-verde mientras que, los pacientes con deficiencias en dicho eje, se confundirán con los colores adyacentes. Los puntos coloreados que conforman los números son isocromáticos, de modo que no pueden distinguirse únicamente por diferencia de contraste. [6]

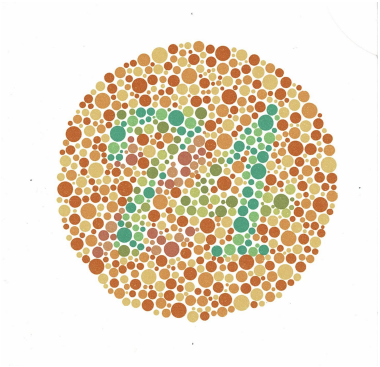

Figura 1.10: Lámina de Ishihara para diagnóstico de protanopia.

1.4. Planificación, tareas realizadas y alcance

La planificación se ha realizado de forma que se distinguen tres fases primarias en el trabajo realizado. Estas fases se corresponden a groso modo con los capítulos uno al tres.

La primera fase del trabajo consiste en el estudio del medio en el que se va a desarrollar la aplicación. En esta fase se estudia el funcionamiento de las cámaras de los distintos dispositivos, así como sus características, respuesta y compatibilidad con los ajustes necesarios. Una vez finalizada esta fase se habrá decidido cual es el dispositivo más adecuado para las tareas.

La segunda fase se centra en torno al desarrollo del ajuste comparativo automático del color representado en una pantalla. Esto es, el diseño, desarrollo y evaluación del algoritmo de búsqueda de unos valores cromáticos que, representados en dicha pantalla, el color de ésta sea percibido de forma análoga al buscado.

La tercera fase es la aplicación a un problema real: la creación mediante la aplicación de una lámina de Ishihara digital reconstruido con una paleta de color equivalente previamente calculada.

Durante la totalidad del trabajo se desarrolla de forma paralela a las fases previamente expuestas la documentación del proyecto y la memoria final, así como diversas reuniones.

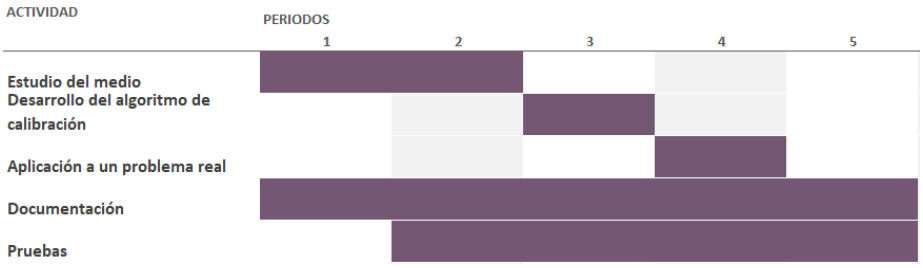

Figura 1.11: Diagrama de Gantt con las tareas del proyecto. Periodos: 1 - Febrero, 2 - Marzo, 3 - Abril, 4 - Mayo, 5 - Junio

El desarrollo de la aplicación, aunque condicionado por la plataforma y el medio, se ha llevado de forma independiente al producto de SmarThings4Vision. Esto quiere decir que la aplicación ha sido diseñada y desarrollada desde cero, salvo algunos módulos descritos a continuación:

- La cámara fue desarrollada en un primer momento a partir del prototipo de código abierto KutCamera. KutCamera es una cámara básica basada en la API Camera de Android. Por razones que se explicarán más adelante, fue necesario migrar a la API Camera2, por lo que casi todo el código de KutCamera fue sustituido en el proceso.
- El módulo de conexión entre los dispositivos utilizados fue proporcionado por SmarThings4Vision. Esta conexión se realiza a través de WiFi mediante el envío de cadenas de caracteres. Al ser suficiente para las necesidades del proyecto, se decidió adaptar esta librería ya desarrollada por la empresa, que será explicada en detalle más adelante.

- Casi todo el procesado de imagen ha sido realizado de cero, pero, en algunas ocasiones, se ha recurrido a la utilización de la librería OpenCV[1]. OpenCV es una librería desarrollada en C++ que contiene multitud de herramientas para el tratamiento de imágenes. Ha sido integrada a través de su distribución para Android y utilizada para procesado de imágenes como *blur* o reducción de ruido.

1.5. Resumen del contenido de la memoria

En esta memoria se describirán los contenidos a lo largo de tres capítulos y finalmente se presentarán las conclusiones del proyecto. En el Capítulo 2 se estudiarán las características de los dispositivos que se utilizarán a lo largo del proyecto. En el Capítulo 3 se detallará el desarrollo del algoritmo de calibración desarrollado. En el Capítulo 4 se explicarán mas en profundidad detalles de implementación de la aplicación. Finalmente, en el Capítulo 5 se mostrarán las conclusiones obtenidas del proyecto y el posible trabajo futuro derivado.

Adicionalmente, en el Anexo A se detallarán las horas dedicadas al proyecto en sus distintas fases y tareas y otros aspectos de gestión. En el Anexo B se podrá consultar un breve manual de usuario sobre el funcionamiento de la aplicación.

[1] OpenCV: `https://opencv.org/`

Capítulo 2

Estudio del medio

En este capítulo se van a tratar los temas relevantes en torno a la toma de imágenes. Se expondrán también las características de los dispositivos probados a tal efecto y de las características que han llevado a su elección.

2.1. Introducción

En primer lugar, para la metodología a desarrollar hacen falta dos dispositivos: uno que funcione como pantalla a calibrar, que de ahora en adelante será llamado *Pantalla*, y otro que controle el algoritmo y haga las veces de *calibrador*, que de ahora en adelante será llamado *Controlador*. Entre ambos dispositivos se establece una relación Maestro-Esclavo donde el *Controlador* asume el rol de Maestro y *Pantalla* el de Esclavo, ejecutando únicamente las acciones que el Maestro le indique. Se contó con los dispositivos de la tabla 2.1 durante la realización del proyecto.

Las funciones de cada dispositivo son las siguientes:

- **Pantalla**: Muestra los colores que el controlador le envíe. Una vez finalizado el algoritmo también será encargado de generar el test de Ishihara con los colores obtenidos.
- **Controlador**: Ejecuta el algoritmo y envía los colores calculados a la pantalla. También dispone de una interfaz de usuario desde donde se pueden manejar los controles de la aplicación. Más información acerca de la interfaz en el Anexo B.

Tabla 2.1: Modelos disponibles durante el desarrollo

Nombre Comercial	Modelo	Pantalla	Cámara	Versión SO
Samsung Galaxy Tab A6	SM - T280	7” (1280x800)	5MP	5.1.1 (API 22)
Samsung Galaxy Tab A6	SM - T580	10” (1920x1200)	8MP	6.0 (API 23)
BQ Aquaris	M10	10.1” (1920x1200)	8MP	5.1 (API 21)
BQ Aquaris	M8	7” (1280x800)	5MP	6.0 (API 23)
BQ Edison	3 Mini	8” (1280x800)	5MP	4.4 (API 19)

Para implementar esas funciones se ha hecho uso de dos elementos hardware clave para el proceso: La cámara del dispositivo *Controlador* y la pantalla del dispositivo *Pantalla*.

2.2. Captura de imagen

La captura de imagen es uno de los temas más delicados del proyecto. Es necesario tener un control preciso sobre los parámetros de la cámara y limitar al máximo los ajustes automáticos de forma que sea posible mantener unas condiciones fijas durante el tiempo que se esté ejecutando el algoritmo. Al tratarse de una aplicación Android diseñada para ejecutarse sobre un dispositivo Tablet, se utilizará la cámara de dicho dispositivo, siempre y cuando permita controlar dichos parámetros.

El control y manejo de la cámara en Android se realiza a través de su API *Camera*, cuya versión 1.0[1] fue abandonada tras la API 21 de Android en la cual se añadió la nueva versión *Camera2*[2]. Aunque pueda parecer una continuación, la API de Camera2 poco o nada tiene que ver con su predecesor, pues supone un cambio de paradigma y una nueva manera de entender el funcionamiento de la cámara. Estas condiciones propician que, en un principio, se comenzase a desarrollar para la API Camera1, pues de esta forma sería compatible para todos los dispositivos ya que, a pesar de encontrarse abandonada, sigue siendo compatible con los nuevos modelos. Para ello se partió de la aplicación de código abierto KutCamera, una aplicación de cámara sencilla basada en la API de Camera1.

Sin embargo, tras probar la respuesta de la cámara y de la pantalla (punto 2.3), se descubrió que la API Camera no permite establecer el tiempo de exposición, sensibilidad y enfoque de forma manual. Tras comprobar que Camera2 sí ofrece control sobre estos parámetros, se decidió migrar la aplicación a dicha API. Aún así, Camera2 depende del grado de compatibilidad, pues no todos los dispositivos pueden hacer uso de todas sus características, siendo éstas dependientes del nivel de compatibilidad proporcionado por el firmware de cada cámara. Es por ello que, pese a que se comenzó trabajando con el dispositivo SM - T280 como controlador, finalmente se tuvo que utilizar el SM - T580 debido a que el primero no es compatible con Camera2, mientras que el segundo lo es con casi todas sus características.

La captura de imagen se realiza de forma continua, por lo que se utilizan las imágenes de la *Preview*, es decir, cada frame que se ve en la pantalla es procesado.

2.3. Respuesta de cámara y pantalla

Otro de los puntos críticos en lo respectivo a los dispositivos utilizados es la respuesta de la cámara y de la pantalla. Tras el análisis de varias imágenes tomadas por la cámara, se observó como la respuesta no es uniforme en intensidad a lo largo de todo el campo. Esto podría ser causado tanto por la pantalla

[1] Camera API: `https://developer.android.com/reference/android/hardware/Camera`

[2] Camera2 API: `https://developer.android.com/reference/android/hardware/camera2/package-summary`

como por el detector de la cámara. Sin embargo, tras tomar varias fotos con distintos enfoques y desplazamientos de la cámara, se observó que el área de distribución de la intensidad se mantiene constante en desplazamiento pero no entre enfoques, por lo que se llegó a la conclusión de que es causado por una aberración óptica debido a las lentes encargadas del enfoque.

Como una primera solución al problema se intentó calcular una matriz correctora que, una vez aplicada sobre una imagen de color constante en todo el campo tomada por la cámara, diera como resultado una distribución uniforme de la intensidad.

Para el cálculo de estas matrices (una para cada canal R, G y B), se ajustó la exposición de cada imagen de forma que se consiguieran valores de intensidad máximos cercanos a 250 y, analizando la distribución de la intensidad en estas imágenes, se pudieron calcular las matrices para la obtención de la salida constante. Al aplicar estas matrices a imágenes tomadas a posteriori, se observó que se comportaba bien cuando se trataba de imágenes lisas compuestas de un solo color primario, pero que cuando se aplicaban a imágenes con colores compuestos este comportamiento no era el esperado.

En un primer momento se pensó en esta matriz correctora como una matriz multiplicativa, por lo que, tras un pequeño análisis de los resultados de este comportamiento, se llegó a la conclusión de que la pérdida de intensidad en la periferia de la imagen llegaba a provocar valores 0 en algunos canales individuales y, debido al carácter multiplicativo de la matriz creada, no era efectiva en esos valores. Este problema se ve representado en la figura 2.1

Se pensó entonces en una aproximación diferente que diese una solución a este problema: crear una matriz aditiva. Esta aproximación resultó un fracaso igualmente, debido a que cuando los valores de intensidad eran bajos en los canales no dominantes, se volvía a detectar su intensidad como 0 pero, debido al carácter aditivo de la matriz, para que esos valores fueran correctos deberían de ser negativos, lo cual no es posible. Esto provoca que se produzca una sobrecompensación en las *colas* de la matriz, como se observa en la figura 2.2

Finalmente, se decidió descartar este ajuste y en su lugar se utilizan recortes situados en torno al centro de la pantalla, donde la respuesta es uniforme como se muestra en la figura 2.3. Estos recortes son de tamaño 100×100 y deben ser ajustados a mano por el usuario, para mayor comodidad y adaptabilidad al entorno. Además, el enfoque se ajusta para que la imagen esté ligeramente desenfocada y cause un *blur* natural, de forma que se evite capturar los píxeles de forma individual.

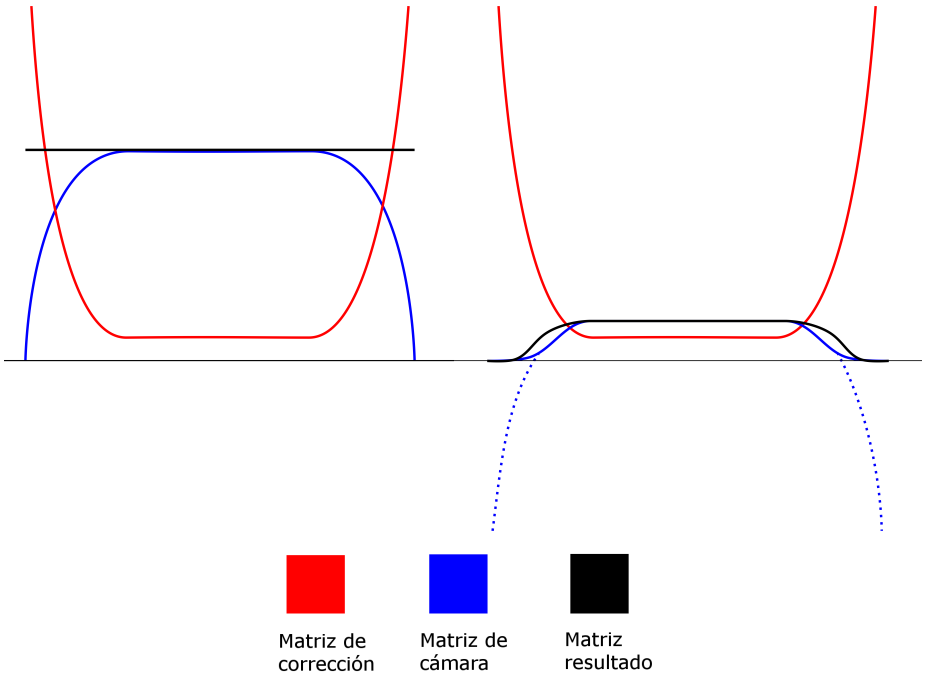

Figura 2.1: Ejemplo de funcionamiento de la matriz multiplicativa con imagen de entrada de valores altos, para la cual se produce una salida correcta (izq.) y con imagen de entrada con valores bajos, para la cual se produce una salida no constante (dcha.).

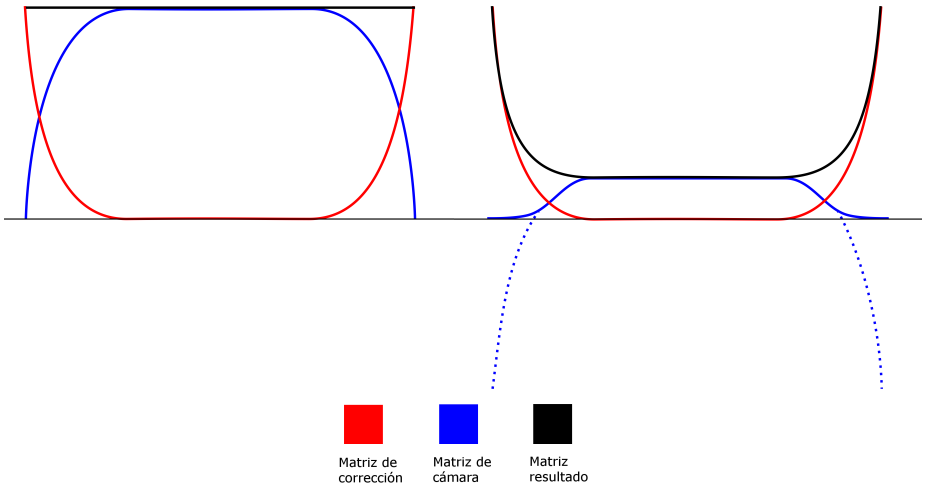

Figura 2.2: Ejemplo de funcionamiento de la matriz aditiva con imagen de entrada de valores altos, para la cual se produce una salida correcta (izq.) y con imagen de entrada con valores bajos, para la cual se produce una salida no constante (dcha.).

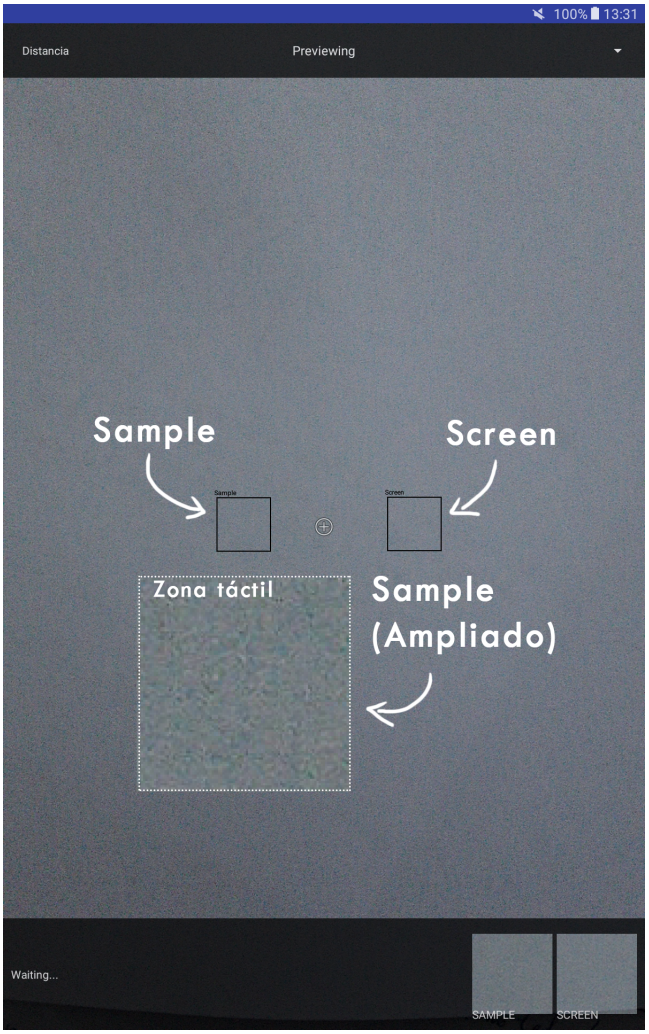

Figura 2.3: Interfaz mostrando los dos recortes móviles utilizados para calcular los colores a comparar, incluyendo la ampliación de uno de ellos.

Capítulo 3

Proceso de calibración del color propuesto

Una vez establecidos los elementos hardware que se han utilizado y la motivación detrás de dicha elección, en este capítulo se tratarán los temas relacionados con el desarrollo y el funcionamiento del algoritmo de calibración desde el concepto inicial hasta la versión final, pasando por las distintas ideas barajadas.

3.1. Terminología

Por la naturaleza del problema, la terminología utilizada puede resultar confusa. Para introducir los términos primero hace falta definir dos **conceptos**:

- **Vector de entrada:** Valores de un vector de color (r, g, b) para ser representado en una pantalla.
- **Vector de salida:** Valores de un vector (r, g, b) recogidos por el detector de la cámara.

Una vez establecidos estos conceptos se puede realizar una breve explicación de los **términos**, que se utilizarán durante el resto del capítulo:

- **Color capturado** o **Color de la pantalla** ($C_{capturado}$): Vector de salida de la pantalla.
- **Color buscado** o **Color de la muestra** ($C_{buscado}$): Vector de salida de la muestra.
- **Color representado** ($C_{representado}$): Vector de entrada de la pantalla.

En la figura 3.1 se puede encontrar una representación de la función de cada término en el sistema.

3.2. Requisitos

Para entender correctamente las funcionalidades que se desean implementar, se listan los requisitos que debe cumplir el proceso de calibración:

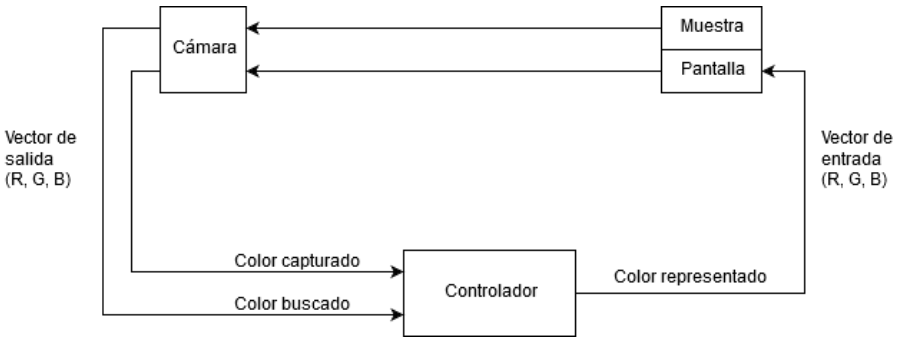

Figura 3.1: Diagrama anotado con los términos explicados en el proceso de calibración. Este proceso es un sistema de re-alimentación que de manera iterativa intenta minimizar la diferencia entre el color buscado y el capturado.

- Debe permitir colocar y mover libremente por la pantalla dos cajas de 100x100, llamadas *SCREEN* y *SAMPLE* que muestrean la pantalla del dispositivo y la muestra, respectivamente.
- La caja *SAMPLE* debe poderse ampliar para seleccionar el color deseado de forma que no todos los objetos tengan que tener un tamaño definido para poderse igualar.
- Debe conocerse en todo momento la distancia vectorial entre el color representado por la pantalla y el capturado en la muestra.
- El color representado por la pantalla debe cambiar automáticamente cuando el controlador alcance una nueva estimación.
- Cuando la distancia entre los vectores de salida sea menor de un valor establecido como aceptable, el algoritmo debe parar e informar de que ha encontrado una solución.

3.3. Prototipado de funcionalidades

Una vez conocidos los requisitos mínimos a implementar se va a proceder a presentar las diferentes funcionalidades desarrolladas y sus aportaciones a la versión final.

3.3.1. Comparación y cálculo de distancias

La comparación entre colores es una distancia euclídea d entre dos vectores de color c_1 y c_2.

$$\begin{aligned} d(c_1, c_2) &= \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \\ c_1 &= (r_1, g_1, b_1) \\ c_2 &= (r_2, g_2, b_2) \end{aligned} \tag{3.1}$$

Durante la ejecución del algoritmo la distancia que se va a manejar es aquella entre el color buscado y el color representado.

3.3.2. Valores iniciales

Una de las partes fundamentales del algoritmo es el punto de partida. Para comenzar a medir y estimar nuevos valores se han de tener dos valores iniciales: Un color buscado y un color representado. El color buscado no puede ser controlado, pero el color representado sí puede ser ajustado antes de comenzar a comparar. Si no se ajustara el color representado inicial, siempre se comenzaría a comparar desde un color negro o del color previo de la pantalla, lo que a menudo resultaría en distancias impredecibles que podrían incrementar la cantidad de estimaciones a realizar para conseguir un resultado aceptable. Esto se puede mejorar realizando una estimación inicial basada en datos grabados de antemano.

Para ello se ha realizado un pequeño estudio donde se mide la respuesta de la pantalla al enviarle colores primitivos rojo, verde y azul en diferentes intensidades. Se ha buscado con este estudio calcular una función inversa aproximada que describa la respuesta de la pantalla a dichos valores. Los resultados se guardan como archivo local y es cargado cada vez que se inicia el programa. Estos valores pueden ser utilizados por dicha función $f(x)$, detallada en la Ecuación (3.2), para obtener un color representado inicial $C_{inicial}^{representado}$ o *semilla* tal que su distancia hasta el color buscado $C_{buscado}$ sea menor que la de un color aleatorio.

$$f(C_{inicial}^{capturado}) = C_{inicial}^{representado} \mid d(C_{inicial}^{representado}, C_{buscado}) < d(C_{aleatorio}, C_{buscado}) \tag{3.2}$$

3.3.3. Colores entrelazados

Durante el estudio desarrollado en el punto 3.3.2 se detectaron unas anomalías que al principio se pasaron por alto: el incremento de la presencia del color verde a niveles altos provoca un incremento en la detección del color azul, siendo éste último inexistente en la fuente original. Este fenómeno no afecta a los colores rojo y azul, como puede observarse en la figura 3.2

Esto dificulta al algoritmo a la hora de encontrar colores con componentes verdes altos dado que, cuando sube la cantidad de verde para igualar, indirectamente se incrementa la detección de azul. Se provoca entonces un estado en el que no es posible diferenciar cuándo existe una presencia de azul real y cuándo es provocada por el verde. Como se detallará en el punto 3.3.5, se alterará la función de estimación para el color azul para que tenga en cuenta el valor del canal verde, de forma que se puedan detectar esos falsos estímulos.

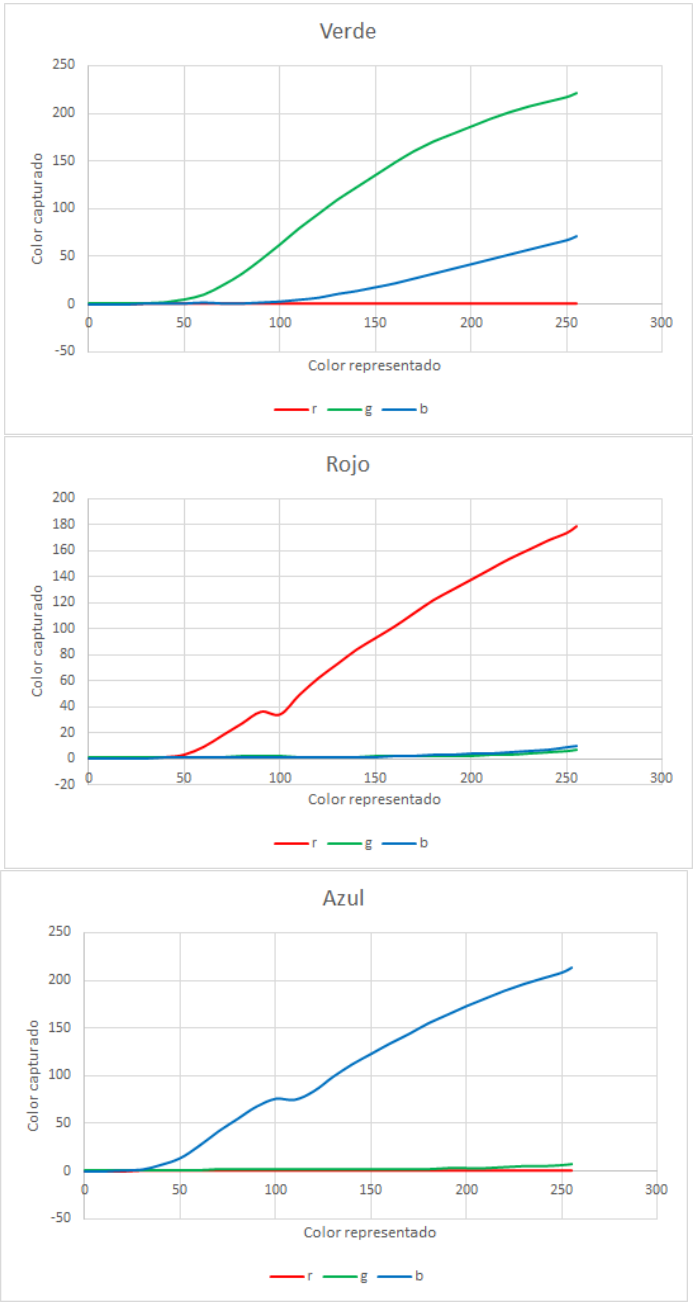

Figura 3.2: Respuesta de la detección de colores de la cámara frente al incremento de los valores de los colores primarios rojo, verde y azul.

3.3.4. Estimación por fuerza bruta

En el primero de los métodos de estimación desarrollado se ha implementado una función lineal basada en la diferencia $diff$ de los valores de cada canal individual (R, G y B) entre el último color capturado $C_{capturado}$ y el color

buscado $C_{buscado}$. A esta diferencia se le aplica un coeficiente multiplicativo b, se le suma un valor de incremento mínimo a y se suma al último color representado $C_{representado}$ para obtener la siguiente estimación, como se detalla en la Ecuación (3.3).

$$\begin{gathered} diff = C_{capturado} - C_{buscado} \\ f(x) = \begin{cases} C_{representado} + (a + b \cdot |diff|) & \text{if } diff > 0 \\ C_{representado} - (a + b \cdot |diff|) & \text{if } diff < 0 \end{cases} \end{gathered} \tag{3.3}$$

Este método depende de ajustar los coeficientes a y b para definir su precisión. Con valores bajos como $a = 2$ y $b = 0{,}1$, se consigue una método efectivo pero no eficiente, ya que realiza saltos pequeños y requiere de muchas iteraciones para llegar a un resultado aceptable. Por otro lado, si se utilizan valores más altos pueden lograrse resultados más rápidos pero a cambio se pierde mucha precisión debido al incremento de tamaño en los saltos, por lo que se decide dejar los valores previamente mencionados.

3.3.5. Estimación por regla falsa

Se ha utilizado el método de regla falsa como base para el desarrollo de una función de estimación iterativa, es por eso que se ha denominado como *estimación por regla falsa*[7]. Al igual que el método de fuerza bruta, todas las estimaciones se realizan en cada canal por separado, aunque no se especifique durante el texto por razones de claridad.

El problema al que nos enfrentamos es que para una función $f(x)$ queremos encontrar un valor x tal que su y sera igual a $C_{buscado}$. Esta función $f(x)$ representa la relación de $C_{representado}$ con $C_{capturado}$ (Eq. 3.4).

$$f(C_{representado}) = C_{capturado} \tag{3.4}$$

Sin embargo, esta función no es constante, es decir, varía debido a las fluctuaciones provocadas por el ruido de la cámara. Aunque consideramos estas fluctuaciones lo suficientemente pequeñas como para dar por buenas las estimaciones, pueden provocar problemas en el funcionamiento de esta función de estimación que veremos más adelante.

Una vez da comienzo la función, conocemos un punto $p_1 = (x_1, y_1)$ gracias a la aproximación inicial descrita en el punto 3.3.2, donde x_1 es $C^{inicial}_{representado}$ e y_1 su $C_{capturado}$ correspondiente. A partir de este punto hemos de calcular si y_1 es mayor o menor que $C_{buscado}$ y encontrar un segundo punto que esté por debajo o por encima, respectivamente.

Para encontrar dicho punto se utiliza una función que, a partir de la distancia $d = y_1 - C_{buscado}$ calcula el valor y_2 tal como se describe en la Ecuación (3.5)

$$\begin{gathered} t = 1 \\ d = y_1 - C_{buscado} \\ y_2 = \begin{cases} y_1 - t \cdot d & \text{if } y_1 \geq C_{buscado} \\ y_1 + t \cdot d & \text{if } y_1 < C_{buscado} \end{cases} \end{gathered} \tag{3.5}$$

En caso de que ese valor de y_2 siguiera sin ser válido, volvería a realizarse la cuenta incrementando el valor de t en 1. Para calcular el x_2 correspondiente a ese y_2 se utiliza una función lineal por partes de la Ecuación (3.6) representada en la Figura (3.3). Esta función se ha creado de forma que sea una representación aproximada de la respuesta de los colores de la pantalla y, aunque no es precisa, en este paso no hace falta precisión, por lo que es suficiente.

$$f(x) = \begin{cases} 0{,}2x & \text{if } x < 60 \\ -60 + 1{,}4x & \text{if } x \geq 60 \end{cases} \tag{3.6}$$

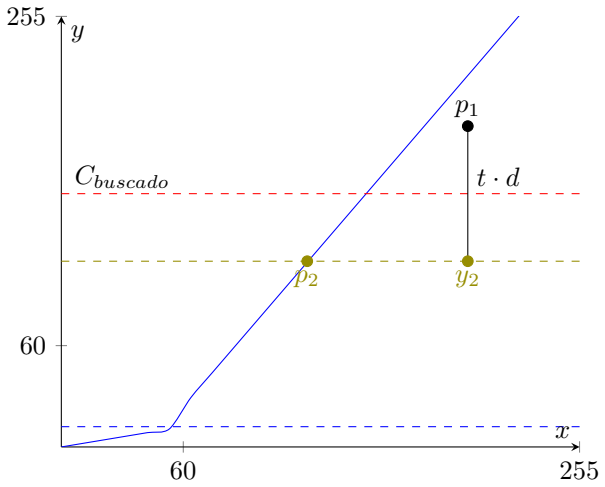

Figura 3.3: Gráfica de la Ecuación (3.6), donde puede observarse representado de forma gráfica el proceso de cálculo del punto p_2: como $x_1 > 60$, se calcula el valor $y_2 = y_1 - t \cdot d$ y se proyecta sobre $f(x)$ para conseguir el valor x_2 tal que $f(x_2) = y_2$, consiguiendo de esa forma el punto $p_2 = (x_2, y_2)$.

Una vez establecidos dos puntos $p_1 = (x_1, y_1)$ y $p2 = (x_2, y_2)$ donde $y_1 > C_{buscado} > y_2$, se traza una línea recta entre ellos que corta a $C_{buscado}$ en un punto x formando un punto p_n. Este valor x será el nuevo $C_{representado}$ que será pintado en *Pantalla* generando un nuevo $C_{capturado}$. Se puede consultar un ejemplo representado en la Figura 3.4.

Por otro lado, para evitar incrementar el valor del azul cuando éste viene dado por el incremento del valor verde (Explicado en el punto 3.3.3), la función de estimación del azul varía ligeramente para depender del valor del verde. Para el color azul, antes de calcular el punto de corte de la recta $\overline{p_1 p_2} = b + a \times x$ con $C_{buscado}$, se aplica una corrección basada en un valor $\beta = 0{,}2$ que representa de forma aproximada dicha dependencia del verde, como se representa en la Ecuación (3.7).

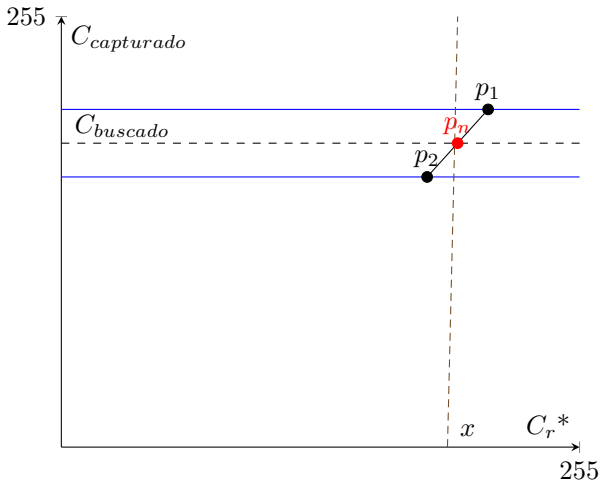

Figura 3.4: Representación simplificada de una búsqueda por regla falsa donde se observa la línea trazada entre p_1 y p_2, cómo corta con el valor de $C_{buscado}$ y se proyecta sobre el plano horizontal para hallar su valor x. $*C_r$ es $C_{representado}$ acortado para facilitar la visibilidad en el gráfico.

$$
\begin{aligned}
G_v &= \text{ Valor del } C_{capturado} \text{ del verde} \\
G_n &= \text{ Valor del nuevo } C_{representado} \text{ del verde} \\
A_n &= \text{ Valor del nuevo } C_{representado} \text{ del azul} \\
\Gamma &= b - \beta \cdot G_v \\
A_n &= \frac{C_{buscado} - (\beta \cdot G_n) - \Gamma}{a}
\end{aligned}
\tag{3.7}
$$

Ahora pueden ocurrir dos escenarios:

$$
\begin{aligned}
\Delta &= \text{ Distancia máxima aceptable} \\
d(C_{buscado}, C_{capturado}) &\leq \Delta \qquad (3.8) \\
d(C_{buscado}, C_{capturado}) &> \Delta \qquad (3.9)
\end{aligned}
$$

Si nos encontramos en el escenario de la Ecuación (3.8), ya tendríamos un resultado correcto y el último $C_{representado}$ sería el color que, representado sobre la pantalla, es equivalente a $C_{buscado}$. Por el contrario, si nos encontramos con el escenario de la Ecuación (3.9), la aproximación no ha sido suficiente y se debe continuar. El siguiente paso es comprobar si el nuevo $C_{capturado}$ es mayor o menor que $C_{buscado}$, sustituir el punto correspondiente $p1$ o $p2$ por la nueva tupla $p_n = (C_{representado}, C_{capturado})$ y volver a repetir hasta encontrar un valor aceptable.

Sin embargo, es posible que se produzca un escenario donde no se alcance ninguna solución aceptable si $C_{buscado}$ deja de encontrarse entre los puntos p_1 y p_2. Este escenario viene dado por las fluctuaciones provocadas por el ruido de la cámara y son inevitables. Sin embargo, el método de regla falsa sirve para acercarnos bastante a la solución incluso si no es capaz de llegar a ella. Al fallar

cuando la distancia ya no es muy grande permite que, utilizando el método de fuerza bruta de forma complementaria, asegure con casi total seguridad que se encontrará un resultado en un tiempo aceptable.

3.3.6. Ajuste del tiempo de exposición

Para asegurar que no se produzca una sobreexposición o subexposición que provoque la pérdida de información, se debe ajustar el tiempo de exposición al comienzo del algoritmo y mantenerse constante durante la duración del mismo. Para ello se calcula la media del color capturado y, posteriormente, se aplica una transformación del espacio de color RGB al HSV, del cual únicamente nos interesa conocer que su valor V equivale al brillo. Se ajustará la exposición de forma recurrente hasta que dicho valor se encuentre dentro de un rango definido por unos valores mínimo y máximo considerados aceptables.

$$
\begin{gathered}
v = brillo(C_{capturado}) \\
x = \text{tiempo de exposición} \\
f(x) = \begin{cases} x + x \cdot 0{,}1 & \text{if } v < min \\ x & \text{if } v > min \text{ AND } v < max \\ x - x \cdot 0{,}1 & \text{if } v > max \end{cases}
\end{gathered} \tag{3.10}
$$

3.4. Versión final

Una vez explicadas las funcionalidades desarrolladas, falta por ver como encajan entre sí para formar el algoritmo de calibración de color. En la figura 3.5 se puede observar un diagrama de estados representando el funcionamiento final del algoritmo.

Tras varias pruebas y evaluaciones, se descubrió que la estimación por regla falsa es muy efectiva para reducir grandes distancias en un periodo muy corto de tiempo, pero no suele resultar precisa por el ruido de la cámara. Sin embargo, tenemos el método de estimación por fuerza bruta de la sección 3.3.4, que con los valores bajos establecidos es muy preciso, aunque lento. Una combinación de ambos métodos cubre todas las necesidades. Por ello, finalmente se decide que si la estimación de color por regla falsa se queda sin recursos o tarda demasiado fluctuando entre distintos valores, se cambiará al método de estimación por fuerza bruta, que resulta ideal para cubrir esas pequeñas distancias.

El algoritmo comenzará en el estado E0, a la espera de que se le mande empezar. Una vez de comienzo el proceso se pasará al estado E1, donde se mantendrá hasta que el usuario elija el color que desea igualar. Tras seleccionar el color, se pasará a ajustar la exposición (E2) tal y como se ha explicado en el punto 3.3.6. Tras encontrar una exposición adecuada, se avanzará al estado E3 para calcular el valor del color inicial (punto 3.3.2). Una vez calculado este color se pasará al estado E4: la búsqueda de los colores iniciales, donde se mantendrá hasta que encuentre los pares adecuados, tras lo cual pasará al estado E5: la estimación del color por regla falsa. Tanto el estado E4 como el E5 están explicados en la sección 3.3.5. Finalmente, si se consigue encontrar un color cuya distancia con el buscado sea considerada aceptable, se vuelve al estado inicial E0; en caso contrario, si se supera un número máximo de intentos, se utilizará la estimación de color por fuerza bruta (representada en el estado E6) como

último recurso. Después del estado E6 solo puede pasarse al estado inicial E0, ya sea porque se ha encontrado un color aceptable o porque se ha superado una vez más el número máximo de intentos y se considera un proceso fallido. Este máximo de intentos se establece en 20 iteraciones para evitar bucles infinitos en colores que no sean alcanzables por problemas derivados de los colores entrelazados explicados en el punto 3.3.3.

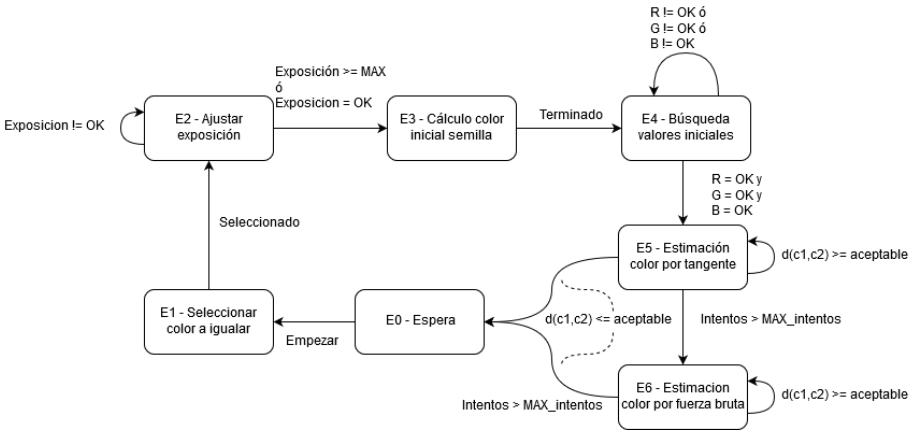

Figura 3.5: Diagrama de estados del algoritmo de calibración desarrollado.

3.5. Evaluación

Para la realización de pruebas sobre el algoritmo se decidió dividir la pantalla del dispositivo *Pantalla* en dos partes iguales, de forma que una mitad muestre un color fijo que simule la muestra y la otra mitad siga funcionando como *Pantalla* como se muestra en la figura 3.6. Con esto se aseguró no tener que cambiar la muestra a mano cada vez que se quiera realizar una ejecución, ya que para cambiarla es tan sencillo como ordenar al dispositivo que cambie el color de la mitad que muestra el color fijo.

Aunque pudiera parecer que los calibradores de pantalla mencionados en el punto 1.3 realizan la misma función, realmente no son comparables porque el objetivo es distinto. Un calibrador de pantalla asegura que esa pantalla se adhiere a un espacio de color concreto, de esta forma es posible representar en dos pantallas distintas los mismos colores. Sin embargo, el objetivo del algoritmo desarrollado es el de obtener una percepción equivalente entre un medio físico y otro digital para casos concretos como el test de Ishihara utilizado en este proyecto. Es por ello que no es posible comparar resultados con calibradores de color ya existentes.

Durante la evaluación se tuvieron en cuenta una serie de variables interesantes para observar sus resultados. De esta forma, la **batería de pruebas** de 50 repeticiones del algoritmo con color de muestra simulado y aleatorio evalúa: si el algoritmo finaliza con acierto o fallo, el tiempo que dura su ejecución y la distancia final entre los colores buscado y capturado. Se considera acierto si la distancia final es menor o igual que un valor Δ previamente definido (Eq. (3.8))

y fallo si esa distancia es mayor (Eq. (3.9)). En este caso, se ha establecido $\Delta = 2{,}0$.

Efectividad del algoritmo. Para comprobar la efectividad del algoritmo se analizan los resultados de la batería de pruebas mencionada. Éstos se pueden observar en la figura 3.7. Se produjo un único resultado desfavorable, del que cabe mencionar que el color a igualar era muy oscuro por lo que no pertenece a priori al abanico de colores para el que se ha diseñado el algoritmo. Por otro lado, hizo falta el uso de fuerza bruta en 37 de las 48 ocasiones en las que se consiguió un resultado favorable, dejando en solo 11 las pruebas en las que se consiguió únicamente con el método de regla falsa.

Eficiencia del algoritmo. Para demostrar la eficiencia del algoritmo se ha analizado el tiempo que tarda en encontrar un resultado y cual es la distancia a la que ha finalizado, correctamente o no. Para la batería de pruebas mencionada anteriormente, el tiempo medio fue de 39,64 segundos por prueba y la distancia media fue de únicamente 1,78 (Recordemos que la distancia se mide como se explicó en el punto 3.3.1).

Con el objetivo de mostrar más datos para poder comparar, en las figuras 3.8 y 3.9 se pueden observar los tiempos y distancias de versiones anteriores del algoritmo v0.9 y v1.0, además de la versión final v1.1. En ellas se pueden observar algunos datos relevantes como el menor tiempo final en la versión v0.9, causado por un error que provocaba la parada del algoritmo antes de tiempo, que a su vez provocaba una mayor cantidad de fallos y, por tanto, aumentó la media de la distancia hasta $10,75$. En la versión v1.0 se solventó dicho error, aunque de una manera brusca que incrementaba el tiempo de ejecución y fue mejorada para la versión v1.1 y final.

Estos datos arrojan un resultado positivo y se considera que la implementación es exitosa.

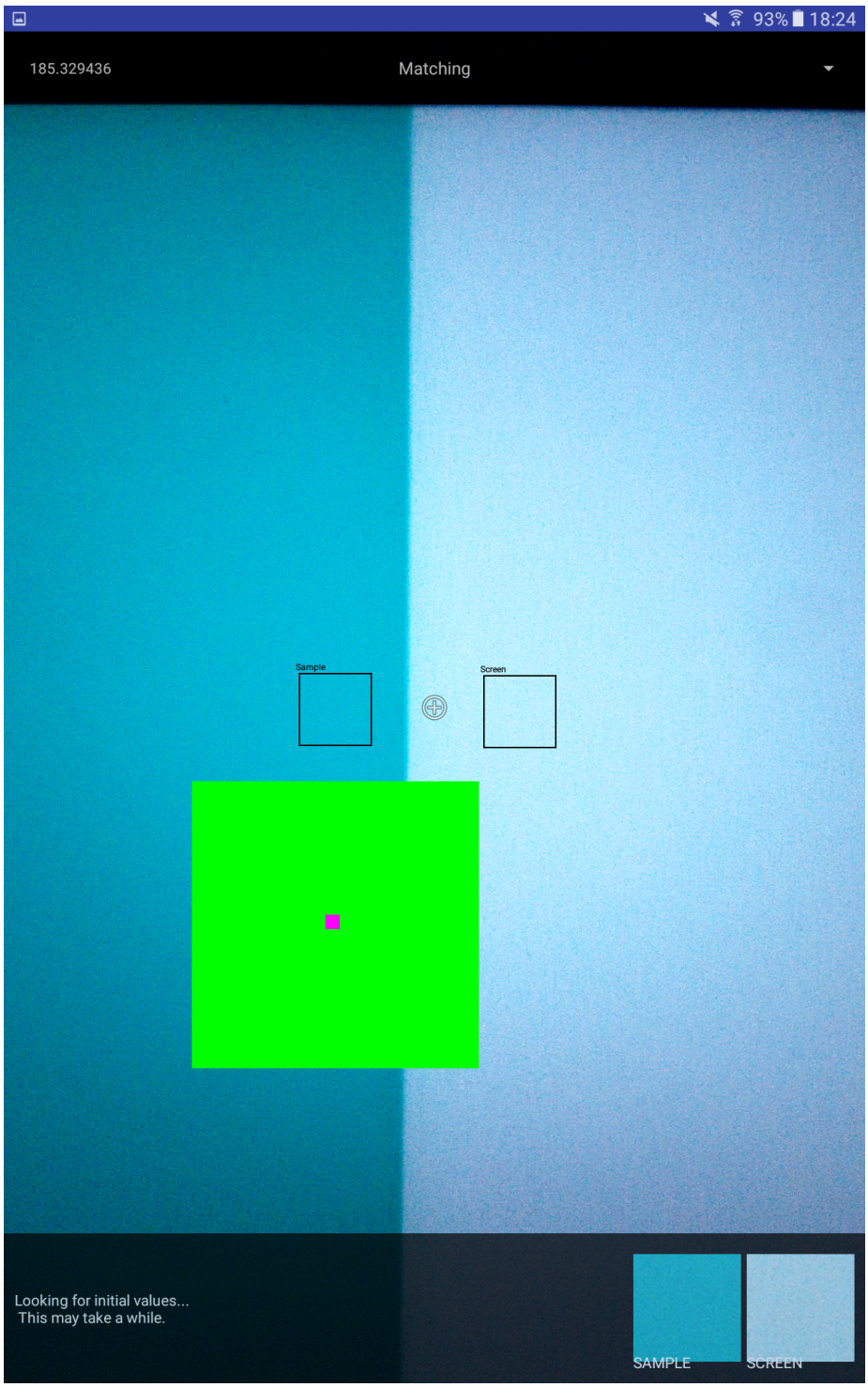

Figura 3.6: Captura de pantalla que muestra la división de la pantalla desde el punto de vista del *Controlador* durante la ejecución del algoritmo.

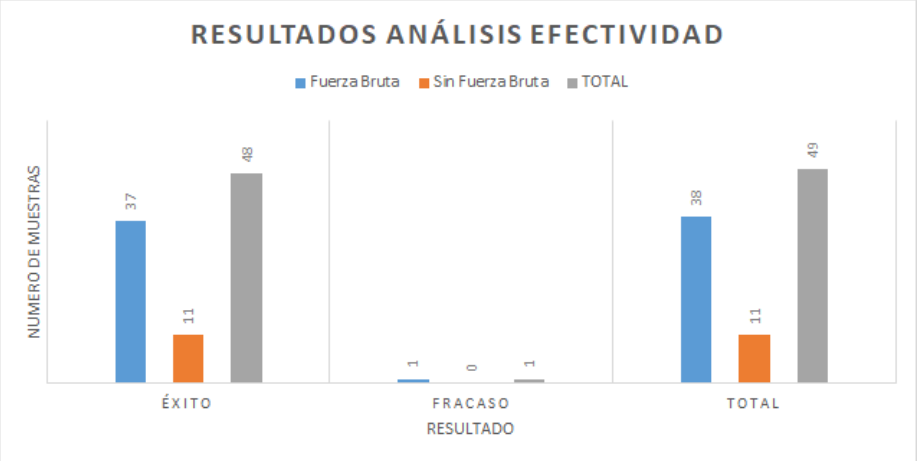

Figura 3.7: Resultado del análisis de efectividad del algoritmo de calibración desarrollado a partir de los resultados de la batería de pruebas.

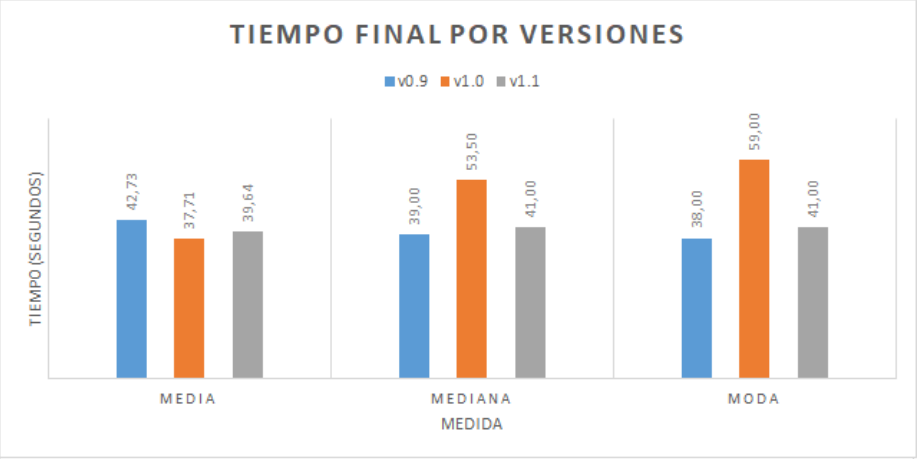

Figura 3.8: Resultado del estudio del tiempo de ejecución del algoritmo de calibración desarrollado en distintas versiones a partir de baterías de pruebas.

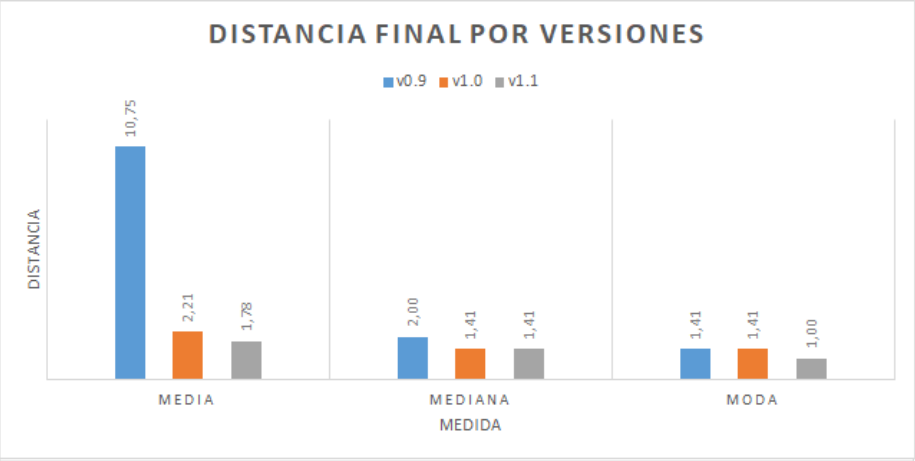

Figura 3.9: Resultados del estudio de la distancia entre $C_{buscado}$ y $C_{capturado}$ al final del algoritmo de calibración desarrollado en distintas versiones a partir de baterías de pruebas.

Capítulo 4

Diseño y evaluación del sistema propuesto

En este capítulo se van a tratar los temas relacionados con el desarrollo de la aplicación, los componentes hardware, sus conexiones y pruebas realizadas para comprobar si el funcionamiento es correcto.

4.1. Componentes hardware del sistema

Aunque ya sabemos de la existencia de los dos componentes protagonistas *Controlador* y *Pantalla*, también hay que tener en cuenta otros aspectos como la conexión entre dispositivos o la luz ambiente.

Conexión. Para poder transmitir el color representado del dispositivo *Controlador* al dispositivo *Pantalla*, éstos se han de haber conectado previamente. Los únicos requisitos del sistema de conexión son: que la conexión sea inalámbrica y que permita el envío de cadenas de caracteres a través de mensajes.

Una conexión inalámbrica no resulta un problema, ya que los dispositivos nunca están muy alejados entre sí, por lo que podría utilizarse tanto Bluetooth como WiFi. Afortunadamente, OptoTab, el producto de SmarThings4Vision, tuvo estos mismos requisitos y cuenta con un sistema de conexión WiFi que cumple con lo requerido por nuestra aplicación. Es por ello que se decidió usar el mismo sistema utilizando el paquete implementado por SmarThings4Vision e integrándolo en el sistema.

Este sistema consiste en que el dispositivo esclavo, en este caso el *Pantalla*, genere una red Wifi como *Hotspot* a la que el dispositivo maestro, nuestro *Controlador*, se conecta para enviar mensajes.

En la figura 4.1 se puede observar un diagrama de clases representando el paquete utilizado, su jerarquía y la implementación en el sistema. Para ello, nuestro **MainActivity** hace uso de WifiHotspotManagerNew, tanto para gestionar la creación del *Hotspot* como para conectarse y enviar mensajes. Además, hizo falta implementar el método *OnGettingMessage* de WifiApManager para gestionar la recepción de mensajes.

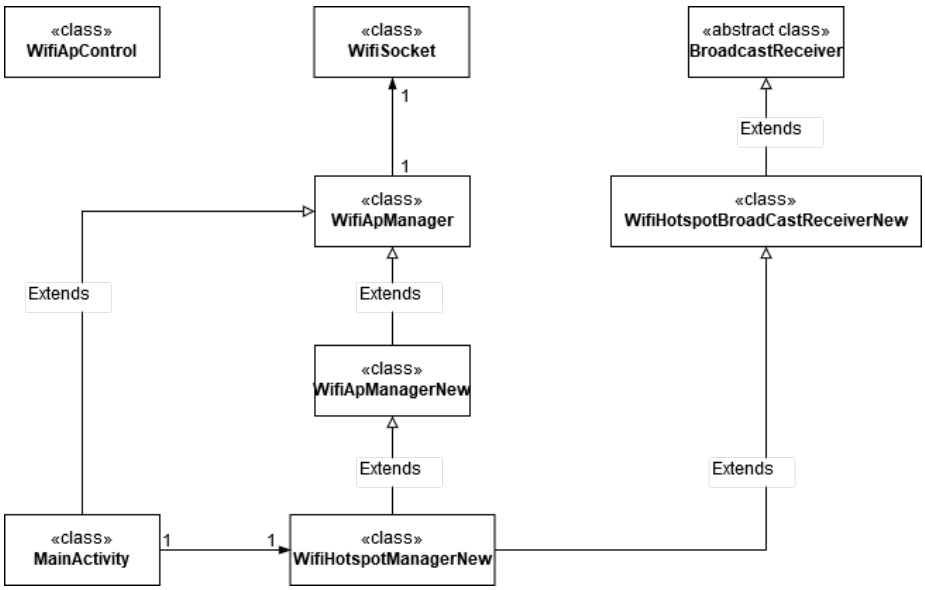

Figura 4.1: Diagrama de clases del paquete de conexión.

Luz ambiente. La luz ambiente juega un papel importante en el funcionamiento de la aplicación debido ya no solo a los posibles reflejos sino al efecto que tiene sobre el color buscado (recordemos el punto 1.3).

Tanto es así, que para las pruebas del algoritmo de la sección 3.5, donde se deseaba la menor luz ambiente posible que pudiera alterar los resultados, se utilizó una caja para bloquear todo tipo de luz externa, dejando únicamente un espacio para encajar la cámara del *Controlador*.

Sin embargo, a la hora de realizar pruebas con objetos reales sí es necesaria la existencia de una fuente de luz externa, por lo que no puede seguir utilizándose la caja. Se pensaron otras soluciones como la creación de una caja que permitiera pasar la luz a a través de un difusor, que incluso se llegó a construir, aunque finalmente se llegó a la conclusión de que era mejor realizar el proceso con la misma luz ambiente, para que los colores se igualasen bajo las mismas condiciones en las que después se fuera a utilizar. Al ser un producto enfocado a un uso en gabinetes optométricos, cuyo ambiente lumínico está altamente controlado y no suele variar, se considera una solución perfectamente válida.

4.2. Aplicación.

La aplicación puede representarse como componentes de alto nivel relacionados entre sí, de forma que quede una vista fácilmente comprensible del sistema. Estos componentes, que pueden encontrarse en la figura 4.2, son:

- **OpenCV:** Librería de OpenCV para Android[1], desarrollada en C++ pero accesible en Java a través de una interfaz Java JNI. Aparece como depen-

[1]OpenCV for Android: `https://opencv.org/platforms/android/`

dencia en otros componentes ya que es utilizado para la manipulación de imágenes como el uso de efectos *blur* o *denoise*.

- **AuxiliaryObjects:** Contiene clases de apoyo ya sean clases que encapsulan funciones estáticas u objetos instanciables para almacenamiento y manipulación de datos.
- **Comparator:** Realiza operaciones de comparación de colores y se encarga de manejar los recortes de la pantalla.
- **GUI:** Contiene las Actividades[2] y vistas[3].
- **Connectivity:** Contiene todo el código relacionado con la conexión entre dispositivos, como se explicó en el punto 4.1.
- **IshiharaTest:** Encapsula la información y operaciones necesarias para representar un test de Ishihara.

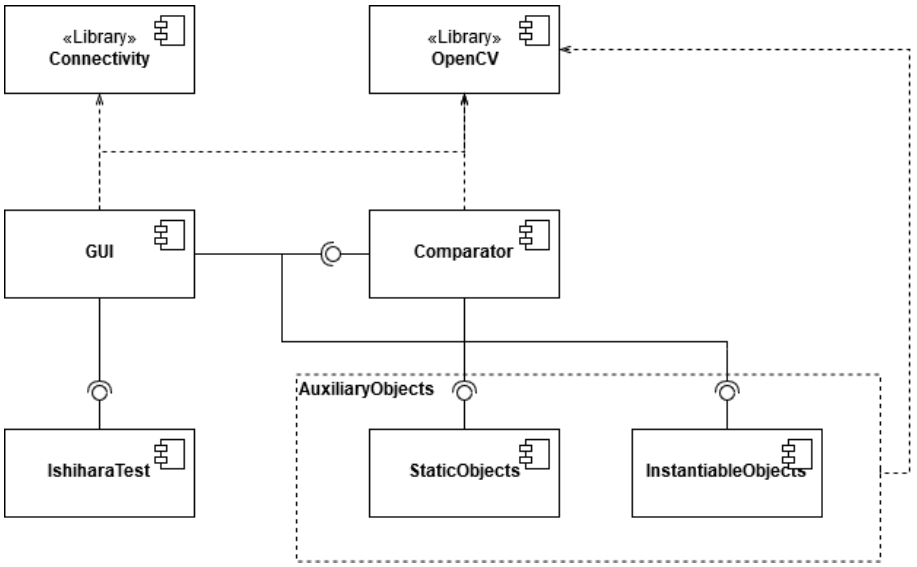

Figura 4.2: Diagrama de componentes de alto nivel de la aplicación.

Aunque el código es el mismo, la aplicación reconoce si el modelo sobre el que se está ejecutando es *Pantalla* o *Controlador* para crear la interfaz correspondiente. En la figura 4.3 pueden verse las interfaces de *Controlador* y *Pantalla* respectivamente.

La interfaz de *Pantalla* consta únicamente de una superficie lisa sobre la que se pinta un color a pantalla completa o partida, dependiendo de la orden que le llegue.

La interfaz de *Controlador* es más compleja, aunque consta únicamente una pantalla desde donde se pueden manejar todos los controles a través de un

[2]En Android, una Actividad (Activity) es la clase que encapsula y relaciona una pantalla de la GUI y el código necesario para su funcionamiento.

[3]En Android, una vista (View) es un componente de la GUI.

menú desplegable en la esquina superior derecha. También muestra información como el estado de la aplicación en la parte superior central, la distancia entre los colores comparados (cuando se esté comparando) en la esquina superior izquierda, el estado del algoritmo (cuando se esté ejecutando) en la esquina inferior izquierda y el contenido de las cajas seleccionadas en la esquina inferior derecha.

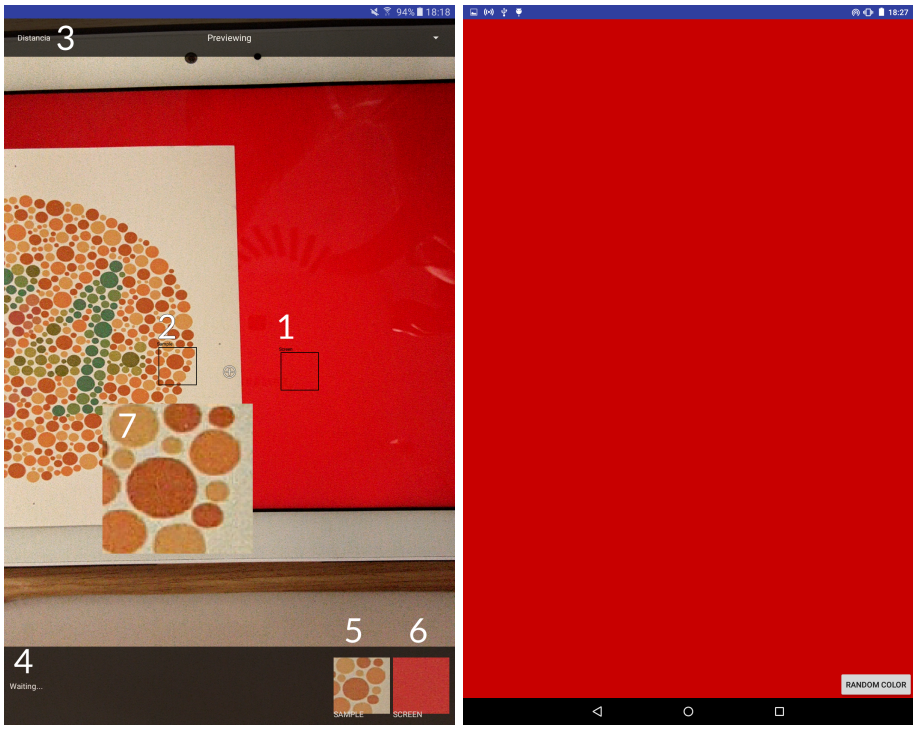

Figura 4.3: Captura de la pantalla de *Controlador* (izq.), mostrando la imagen capturada por la cámara, los recortes para tomar muestras (1 y 2), la distancia (3), el estado del algoritmo (4), la vista previa de los recortes (5 y 6) y la zona ampliada del recorte de la muestra (7). Captura de la pantalla de *Pantalla* (dcha.) mostrando un color liso.

4.3. Verificación

Después del análisis cuantitativo del algoritmo realizado en la sección 3.5, donde quedó demostrado que el algoritmo tiene un comportamiento adecuado, se presentan los resultados cuantitativos de la integración de éste en un problema real. Como aplicación real donde probar el programa desarrollado se eligió el test de Ishihara, explicado previamente en el punto 1.3. Es un test que requiere de ciertas condiciones lumínicas, contiene diferentes paletas de colores y por último, al ser un test, puede verificarse si su comportamiento es correcto comparando los resultados con su equivalente en papel.

Para adaptar la aplicación a la creación de este test se tuvieron que realizar algunas modificaciones. Esta nueva funcionalidad añade unos nuevos requisitos:

- Se debe poder crear y generar láminas del test de Ishihara a través de un proceso del algoritmo de calibración con tantas iteraciones como colores compongan el test.
- Una vez finalizado el proceso de creación, se deben guardar los datos en un fichero para posteriores usos.
- En caso de que exista un fichero con datos, se debe poder generar una lámina sin volver a correr todo el proceso.

En primer lugar, se pensó en como realizar la sustitución de los colores de las láminas del test de Ishihara respetando al máximo su composición original. Para ello, se decidió que lo mejor era separar cada lámina en capas a través de software de edición como GIMP[4]. Cada capa separada contiene únicamente los círculos que corresponden a su color. Por restricciones de tiempo y dificultades en el proceso, solo se pudo realizar esta tarea para una única lámina. Una vez separado en capas, se decidió utilizar un filtro de color para sustituir programáticamente el color original por el nuevo color calibrado (color representado).

En cuanto al proceso, se ha realizado de forma sencilla, dando tiempo al usuario a mover la lámina para poder seleccionar el color a calibrar en caso de que fuera necesario e informando en todo momento de qué color debe calibrar. Esto es debido a que el programa no tiene una forma de reconocer qué color va con qué capa, por lo que se debe seguir un orden estricto programado en el sistema.

El proceso quedó como se observa en el diagrama de estados de la figura 4.4, donde se comienza en el estado de reposo E0. Cuando el usuario ordene que desea crear un Ishihara, primero el programa le preguntará qué lámina desea generar, pasando al estado E1. Una vez seleccionado el test a generar, se pasará al estado E2, donde se mantendrá hasta que el usuario seleccione el color a igualar haciendo click en la zona táctil de la caja *SAMPLE* (Figura 2.3), a la que se aplicará un suavizado (*blur*) para reducir el trazo del pigmento. Tras seleccionar el color, da comienzo la ejecución del algoritmo, representado en el estado E3. Cuando finalice la calibración del color, existen dos opciones: Si era la última capa, se pasa a generar la lámina de Ishihara recién creada (E4), en caso contrario, se retornará al estado E2 tras almacenar el resultado. Cuando termina de generarse la lámina, se mostrará una pantalla como la mostrada en la Figura 4.5 correspondiente al estado E5, desde donde se volverá al estado inicial E0 cuando el usuario desee cerrar el test generado.

4.3.1. Pruebas de campo

Se realizaron algunas pruebas de campo con usuarios con problemas conocidos en la percepción del color. Para ello, se les enseñó la lámina generada en el dispositivo *Pantalla* tras lo cual se les preguntó qué figura veían. Tras ello, se les presentó la lámina física y se les repitió la misma pregunta. Una vez realizadas ambas pruebas, se les presentaron ambas láminas y se les pidió valorar de 0 a 10 cómo de definida veían la figura en cada una de ellas. Por restricciones de tiempo, las pruebas se realizaron en un lugar distinto de donde fue calibrado el dispositivo, por lo que era esperado que los resultados del test digital no fueran

[4]GIMP: `https://www.gimp.org/`

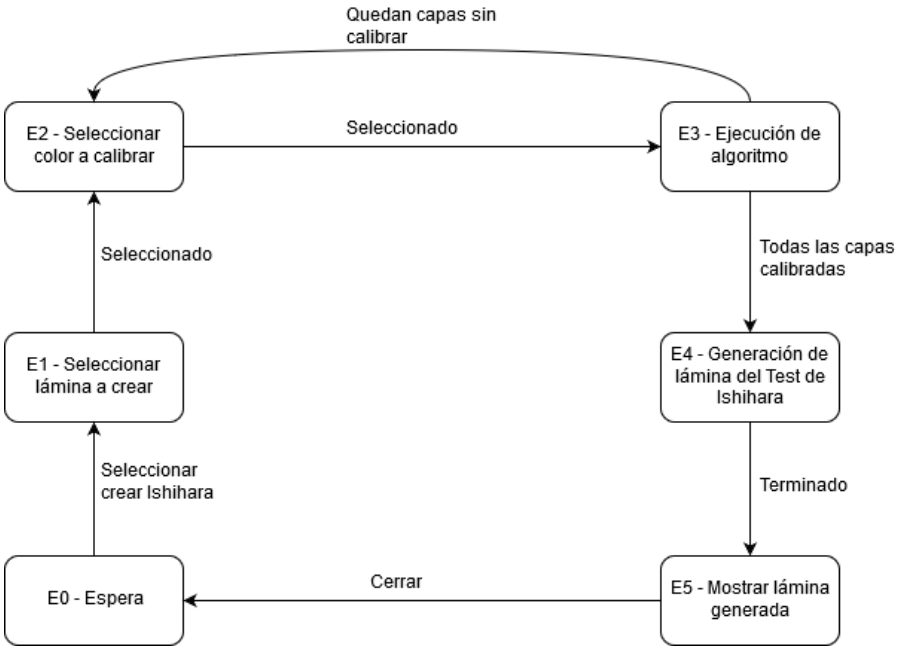

Figura 4.4: Diagrama de estados del proceso de creación de una lámina del test de Ishihara.

del todo satisfactorios. La lámina presentada contiene una figura que las personas sin anomalías perciben como un 74, mientras que las personas con anomalías rojo-verde, como los sujetos de esta prueba, perciben como un 21 y las personas con anomalía cromática total no perciben ninguna figura.

Se realizó la prueba a cuatro sujetos anónimos llamados sujeto A, B, C y D. La prueba del sujeto A se realizó en un entorno parecido al original, mientras que la prueba de los sujetos B, C y D se realizó en unas peores condiciones. Todos ellos reportaron ver en ambos test el número 21, tal y como se esperaba debido a su anomalía, salvo el sujeto D que no reportó ver ninguna figura en el test digital, lo cual podría ser correcto si sufre una anomalía cromática total como se ha mencionado anteriormente. Cuando se les preguntó por su valoración del 0 al 10, los sujetos B y C coincidieron en valorar con un 6.5 la claridad con la que percibían el número en papel y con un 5 en la pantalla. El sujeto A reportó un 10 en papel y un 8 en pantalla. Finalmente, el sujeto D reportó un 0 en pantalla y un 3 en papel, por lo que se puede afirmar que este sujeto sufre una deficiencia más fuerte en la percepción rojo-verde que el resto, lo cual explica que en el test digital no llegue a apreciar ningún numero. En este sentido, nuestra prueba digital parece realizar un diagnóstico más agudo que la lámina en papel, aunque también podría ser debido a las condiciones lumínicas.

Estos resultados arrojan un resultado previsible: en unas condiciones lumínicas propicias, se valora con números mas altos la claridad con la que se perciben las figuras. Aunque parece constante la valoración de que en la figura se percibe ligeramente peor en la pantalla, estas diferencias reportadas no son elevadas y la efectividad de la lámina se ha comprobado cercana a su equivalente en papel,

por lo que, a falta de realizar una evaluación más correcta y extensiva, podría decirse que a primera vista el proyecto ofrece resultados prometedores, ya que todos los sujetos con deficiencias conocidas en la percepción del color tienen una reacción similar en nuestra lámina digital que con la de papel, y en ningún caso daría un diagnóstico negativo a una persona con deficiencias en la percepción del color, ya que de momento parece que la discordancia está en una diagnosis un poco distinta de la agudeza del problema.

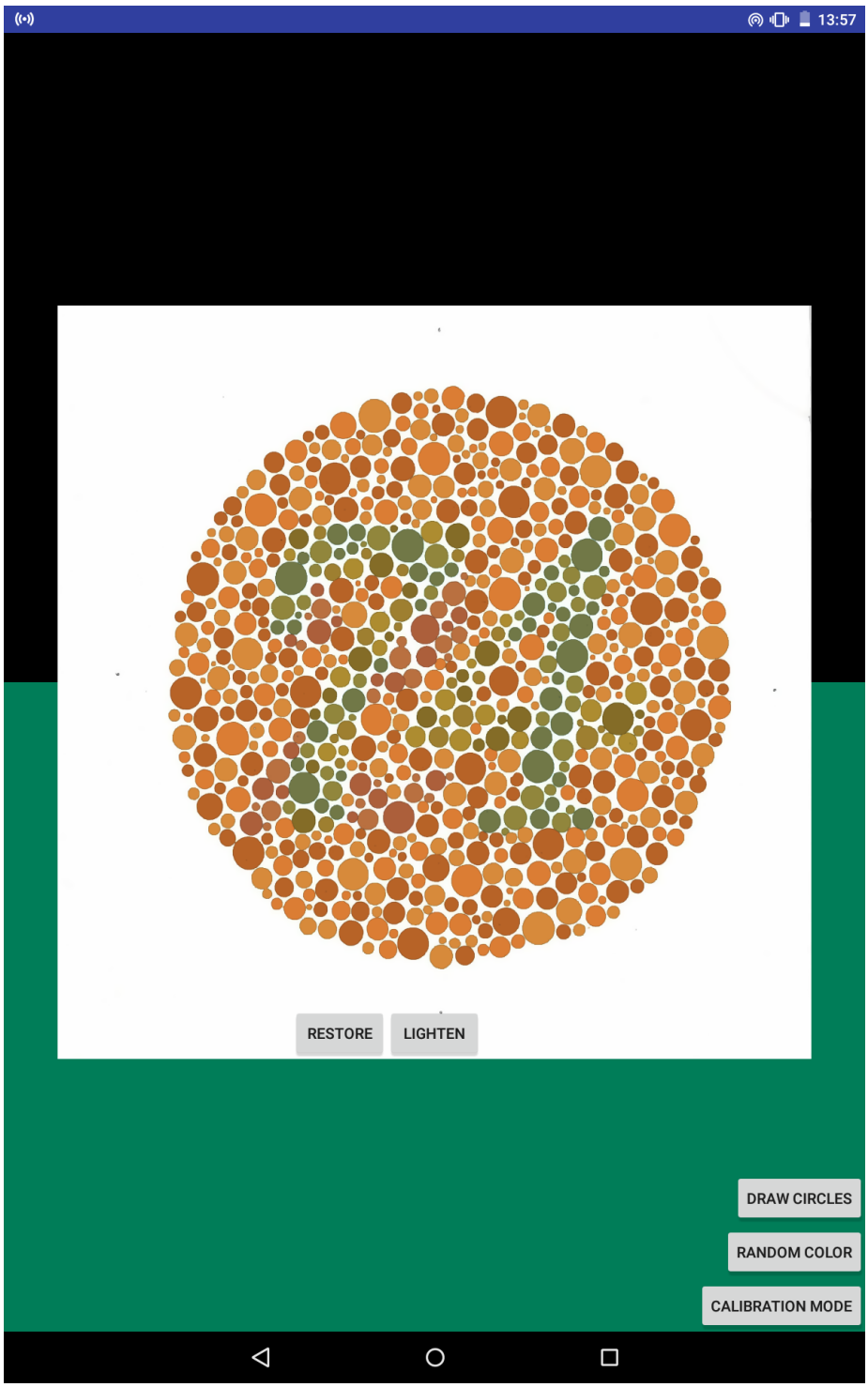

Figura 4.5: Captura que muestra una lámina del test de Ishihara generada en el dispositivo *Pantalla*.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones del trabajo

Este trabajo ha presentado una propuesta para la representación de colores en una pantalla que se perciban de forma equivalente al papel. Este problema se ha atacado mediante la creación de un algoritmo que busque dichos colores y la implementación de una aplicación. Se ha demostrado mediante la recreación de una imagen digital de una lámina del test de Ishihara que el funcionamiento de dicho algoritmo es correcto.

Se ha conseguido crear un test digital para la detección de anomalías en la percepción cromática, cuya efectividad se ha comprobado cercana a su equivalente en papel. Aunque haría falta una evaluación mas intensiva y en mejores condiciones de las que se han impuesto debido a limitaciones temporales y materiales, se puede considerar que la aplicación es viable y podría ser utilizada en un entorno profesional.

En cuanto a los detalles de implementación del prototipo, finalmente está formado por dos dispositivos, uno que hace de pantalla a calibrar y otro que hace de controlador que captura y procesa las imágenes, ejecuta el algoritmo y controla lo que muestra la pantalla. Para el funcionamiento del algoritmo se evaluaron distintas opciones para el cálculo de la estimación como fuerza bruta y regla falsa, aunque ninguna cumplía al completo las expectativas. Es por ello que la mejor implementación y configuración final resultó ser la combinación de ambos métodos, por lo que si regla falsa falla, se aplica fuerza bruta, de esta forma se ha logrado combinar los aspectos positivos de ambos métodos y minimizar los negativos.

5.2. Trabajo futuro

Como se explicó en el punto 1.1, este trabajo tenía como objetivo que se pudiera integrar en el producto de SmarThings4Vision OptoTab y, por tanto, el objetivo futuro más obvio sería el de realizar dicha integración. Previamente, también debería realizarse una evaluación más intensiva de la efectividad de

las láminas de Ishihara generadas, así como integrar nuevas láminas e incluso probar otros tipos de test de detección de anomalías cromáticas.

Otra opción interesante sería la de estudiar el uso de distintos dispositivos con cámaras de más alta gama que permitieran la captura de imágenes con un mayor rango de colores, para comprobar la mejora de la efectividad de este método, ya que la limitación dada por el espacio de color inherente a la cámara se intuye como la mayor limitación en el propio método.

5.3. Conclusiones personales

Al realizar el trabajo durante mi estancia como estudiante becado en SmarThings4Vision, ha supuesto mi primera toma de contacto con el mundo laboral en un ambiente tan privilegiado como el de una SpinOff de la Universidad de Zaragoza. Esto me ha permitido moverme con total libertad, aunque siempre bajo la tutela y supervisión de Fernando. También ha sido mi primera aproximación al mundo de la investigación, ya que gran parte de este trabajo se basaba en supuestos que tuve que investigar y contrastar, además de desarrollar la aplicación.

También he valorado muy positivamente el hecho de que este trabajo no haya tenido demasiada relación con mi especialidad, Ingeniería del Software, pues me ha permitido formarme y ampliar conocimientos en temas tan interesantes como la visión por computador o el procesamiento de imagen.

En general considero que ha sido una muy buena experiencia y que he aprendido mucho ya no solo a nivel académico sino a nivel profesional y personal.

Finalmente, me gustaría agradecer a Fernando su paciencia y disponibilidad para resolver mis dudas en todo momento y a Ana su constante orientación y ayuda.

Apéndice A

Gestión del proyecto

A.1. Gestión de esfuerzos

Se han dedicado un total de 444 horas al proyecto entre investigación, desarrollo, pruebas y documentación. Se comenzó el día 24 de Enero y se ha trabajado a un ritmo constante desde entonces hasta el 27 de Junio, fecha de finalización de la documentación. En el gráfico de la Figura A.1 se observa la evolución de las horas totales acumuladas.

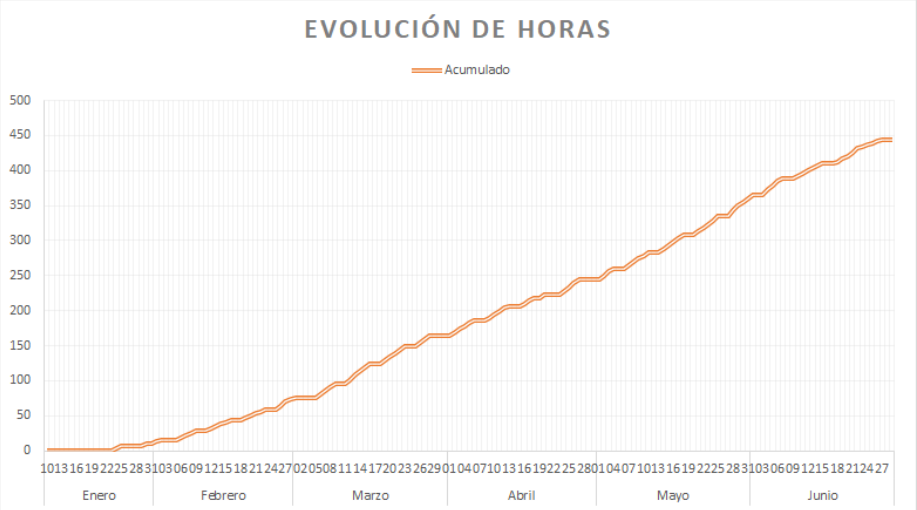

Figura A.1: Gráfico que muestra la evolución de las horas invertidas en el proyecto a lo largo de la duración de éste.

En la Figura A.2 puede observarse más concretamente la cantidad de horas dedicadas en cada mes. Se observa como el ritmo es más o menos regular, con ligeras variaciones.

Otro dato interesante es el reparto de las horas por categorías. Para ello, se han considerado las siguientes categorías:

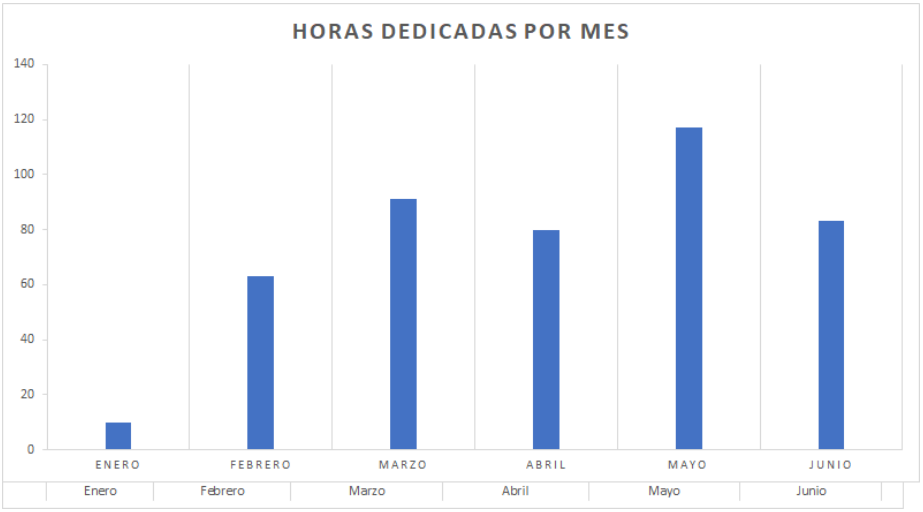

Figura A.2: Gráfico que muestra las horas invertidas al proyecto en cada mes.

- **Investigación y formación**: Entran en esta categoría las horas dedicadas a búsqueda y lectura de documentos relacionados y formación sobre distintos temas.
- **Desarrollo**: Todas las horas dedicadas al desarrollo de la aplicación Android.
- **Pruebas**: Horas dedicadas a testear la aplicación, ya sean las baterías de pruebas mencionadas en el texto o pruebas manuales hechas para comprobar el funcionamiento de nuevas funcionalidades.
- **Documentación**: Aquellas horas dedicadas a la escritura de documentación ya sea de la aplicación o de la memoria.

La categoría a la que más tiempo se ha dedicado es a **Desarrollo** con 134 horas, un 30 % del total. Por detrás, empate técnico entre **Investigación y formación**, **Pruebas** y **Documentación**, con 103, 112 y 90 horas totales que equivalen al 25, 23 y 21 % respectivamente. En general, el tiempo ha estado bien repartido entre las tareas como se observa en la Figura A.3.

También resulta interesante ver la evolución de la inversión de horas en las distintas categorías a lo largo del tiempo como se hace en la Figura A.4. En ella podemos observar una mayor inversión de horas en Investigación y formación en torno al comienzo del proyecto, mientras que las horas invertidas en el desarrollo son más constantes a lo largo del tiempo y aquellas invertidas en pruebas, que comienzan más tarde pero también mantienen un ritmo constante. Por otro lado, las horas dedicadas a documentación no son muchas durante el desarrollo, ya que se documenta únicamente la aplicación y aumentan considerablemente en torno al último mes, cuando se dedica más tiempo a la elaboración de la memoria.

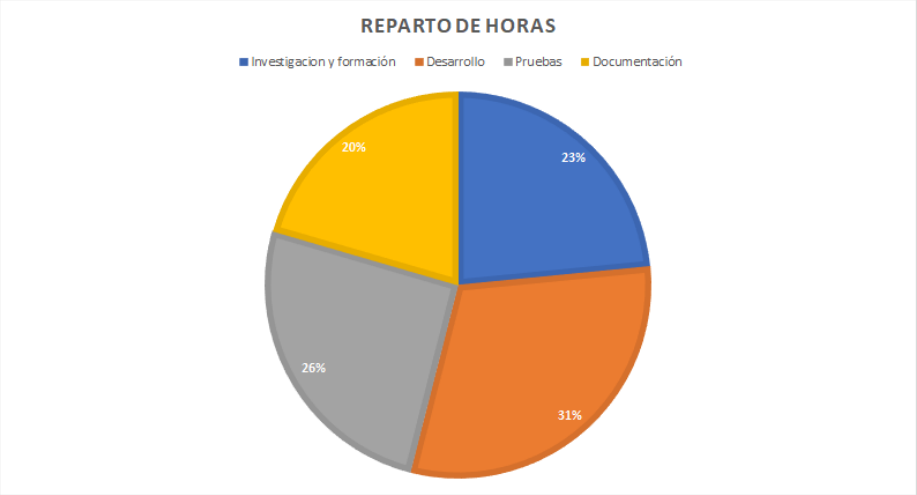

Figura A.3: Gráfico que muestra el reparto de las horas entre las distintas categorías.

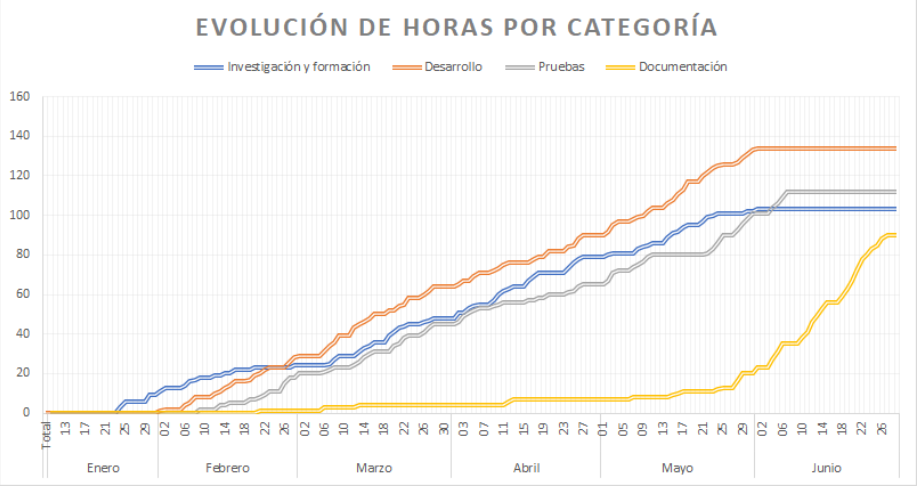

Figura A.4: Gráfico que muestra la inversión de horas entre las distintas categorías a lo largo del tiempo.

A.2. Gestión del código

La versión final de la aplicación suma 8404 líneas de código entre las clases Java desarrolladas, teniendo en cuenta los ficheros XML de la GUI, esta cifra asciende a 11008 líneas de código.

Para el control de versiones se ha utilizado un repositorio privado de Github a lo largo de todo el proyecto, donde se han realizado un total de 45 commits. También se utilizó un repositorio privado paralelo para la gestión de la docu-

mentación con un total de 20 commits.

Apéndice B

Manual de Usuario

En este anexo se incluye un pequeño manual de usuario para explicar el funcionamiento y control de la aplicación desarrollada.

Lanzando la aplicación. La aplicación se encarga automáticamente de crear la red WiFi o conectarse a ella, dependiendo de su rol, pero para asegurar una conexión correcta es altamente recomendable asegurarse de que la aplicación está cerrada en ambos dispositivos y lanzarla primero en el dispositivo *Pantalla*, dado que es el que crea la red, y posteriormente en el dispositivo *Controlador*, dejando unos 4 o 5 segundos de margen. Si la conexión ha sido correcta, veremos un pequeño mensaje *Toast* en la parte inferior de ambos dispositivos confirmándolo como en la Figura B.1.

Elementos de la interfaz Si nos fijamos en la figura B.2 podemos ver los distintos elementos de la interfaz anotados, estos son:

1. **Menú de lanzamiento**: haciendo click aquí se despliega un menú con las operaciones disponibles en la aplicación.
2. **Estado de la aplicación**: informa en todo momento del estado en el que se encuentra la aplicación. Este será distinto dependiendo de la operación que se esté ejecutando, si se está ejecutando alguna.
3. **Distancia**: Aquí se informará continuamente de la distancia entre los colores seleccionados cuando se esté ejecutando el algoritmo.
4. **Caja de recorte de muestra**: esta caja se puede desplazar sobre la muestra para seleccionar el color deseado.
5. **Caja de recorte de pantalla**: esta caja debe colocarse sobre la pantalla para poder comparar con el color de muestra seleccionado.
6. **Caja de recorte de muestra ampliada**: esta caja se desplaza automáticamente al mover la caja de recorte de muestra, contiene lo mismo pero ampliado y permite seleccionar el color a igualar.
7. **Punto de selección de color**: este punto se coloca automáticamente al tocar la zona que se desea seleccionar. El color verde indica la zona del color seleccionado detectada automáticamente.

8. **Estado del algoritmo**: Parecido al estado de la aplicación, informa sobre el estado en el que se encuentra el algoritmo: cargando semilla, buscando color inicial, etc.
9. **Recorte de muestra**: En este cuadro se representa el contenido recortado de la caja de recorte de muestra.
10. **Recorte de pantalla**: En este cuadro se representa el contenido recortado de la caja de recorte de la pantalla.
11. **Centro de la captura**: A modo indicativo, este pequeño dibujo indica el punto central de la pantalla, de forma que se pueda tener en cuenta que es en torno a donde se conseguirán mejores resultados para los recortes

Menú desplegable. Se accede al menú haciendo click sobre la flecha de la parte superior izq (núm. 1 en Fig. B.2), donde despliega una lista con sus opciones como se ve en la Figura B.3. Estas son:

- **Start matching**: Da comienzo a la ejecución del algoritmo de calibración para igualar un único color. Sus resultados no son guardados ni utilizados con ningún fin más que el de demostración del funcionamiento.
- **Stop activity**: Detiene la operación que se esté ejecutando, sea cual sea, y vuelve al estado de espera.
- **Send color**: Despliega un menú flotante que permite introducir un valor personalizado para los canales R, G y B. Una vez introducidos, si el usuario confirma la acción, se envía a la pantalla, donde se pinta a pantalla completa.
- **Send random color**: Envía un color aleatorio a la pantalla, donde se pinta en la media pantalla designada para las pruebas de igualación del algoritmo.
- **Start matching study:** Despliega un menú flotante donde se puede especificar el número de pruebas de igualación a realizar de forma automática. Estas pruebas son realizadas sobre la propia pantalla del dispositivo como se ha explicado en la sección 3.5 y son totalmente automáticas una vez introducido el número deseado. Como no deja de ser un caso de *matching*, se han de haber colocado las cajas de recorte previamente.
- **Build Ishihara:** Despliega un menú flotante donde permite seleccionar la lámina del test de Ishihara que se desea construir. Al seleccionarlo, el icono se pone rojo si el test ya existe en el dispositivo. Si se desea continuar aún así, esos datos serán sobreescritos. Una vez seleccionado, comienza el proceso de construir un Ishihara capa a capa, como se ha descrito en la sección 4.3.
- **Generate Ishihara**: Despliega un menú flotante que permite seleccionar la lámina del test de Ishihara que se desea generar. Al seleccionarlo, el icono se pone verde si el test existe en el dispositivo y rojo si no existe. Si no existe, no puede ser generado. Si existe y se selecciona, el dispositivo *Controlador* enviará a la pantalla la orden para generar dicho test de forma inmediata.

- **Capture screen data**: Proceso de generación de datos para la obtención de la *semilla* de la sección 3.3.2. Este proceso ha de realizarse una única vez antes de poder utilizar el algoritmo y es totalmente automático, únicamente hace falta enfocar a la pantalla de forma que esta forme el total de la pantalla..

Realizando una igualación de color. Para lanzar una igualación de color, primero han de colocarse las cajas de recorte sobre los colores a igualar. La caja de pantalla sobre una zona adecuada de la pantalla y la de muestra sobre el color que se desea seleccionar. Antes de lanzar el algoritmo, ha de seleccionarse el color pulsando en la ampliación sobre el color deseado. Entonces ha de desplegarse el menú (núm. 1 en Fig. B.2) y seleccionar la opción *Start matching*. Comenzará entonces un proceso automático donde se irán actualizando valores en la pantalla como la distancia, el estado de la aplicación y el estado del algoritmo, como se observa en la figura B.4. Cuando el algoritmo encuentre una solución, se detiene e informa al usuario a través de un mensaje en la parte inferior de la pantalla como puede verse en la figura B.5.

Construyendo una lámina de Ishihara. Este es proceso a través del cual es posible generar nuevas láminas del test de Ishihara. Como se ha comentado anteriormente, por restricciones de tiempo únicamente existe una lámina en el sistema, pero está preparado para en cualquier momento poder añadir otras nuevas.

Para comenzar el proceso, al igual que los anteriores, se lanza desde el menú desplegable. En ese momento se desplegará un menú como el de la Figura B.7 en el cual aparecen las distintas láminas recreables. Si seleccionamos una, su icono se pondrá de color verde si esa lámina no existe en el sistema, dado que no hay ningún problema, pero si ya existen datos para la generación de dicha lámina se pondrá de color rojo y aparecerá un mensaje flotante en la parte inferior de la pantalla informando de que esa lámina ya existe y los nuevos resultados sobreescribirán a los antiguos. Se pude cancelar en cualquier momento haciendo click en *Abort*. Para continuar se debe hacer click en *Start*, tras lo cual dará comienzo el proceso tal como se ha explicado en la sección 4.3. El proceso comenzará a igualar el primer color cuando se seleccione a través de la caja de recorte de muestra como se muestra en la figura B.2. Cuando termine de igualar el primer color, la caja de recorte de muestra vuelve a su estado inicial y el proceso vuelve a ponerse en pausa hasta que se seleccione el siguiente color. Debe seguirse el orden establecido en la imagen B.6. El proceso informa mediante el estado de la aplicación qué capa se está generando en cada momento. Cuando se han terminado de generar todas las capas, se guardan los datos de los colores generados en un fichero y se envían a la pantalla para generar la lámina en dicho dispositivo, tal y como se muestra en la figura B.8. Si se cancela en cualquier momento antes de que finalice no se guardarán los datos.

Generando una lámina de Ishihara. En este proceso únicamente se genera la lámina a través de datos previamente generados mediante la creación de la lámina, o mediante colores aleatorios. Para ello se lanza desde el menú la opción *Generate Ishihara*, tras lo cual se mostrará un menú como el de la figura B.9 donde se listan las distintas láminas que se pueden generar. Al seleccionar una

opción, el icono tomará un color verde si se puede generar ese test, en caso contrario, la opción no es seleccionable y se informará a través de un mensaje en la parte inferior de la pantalla. Una vez seleccionado, se puede cancelar haciendo click en *Abort* o continuar haciendo click en *Start*, si se elige esta última, se enviará la orden con los datos a la pantalla para que ésta lo genere. Finalmente, se mostrará en el dispositivo *Pantalla* la lámina generada como en la Figura B.8.

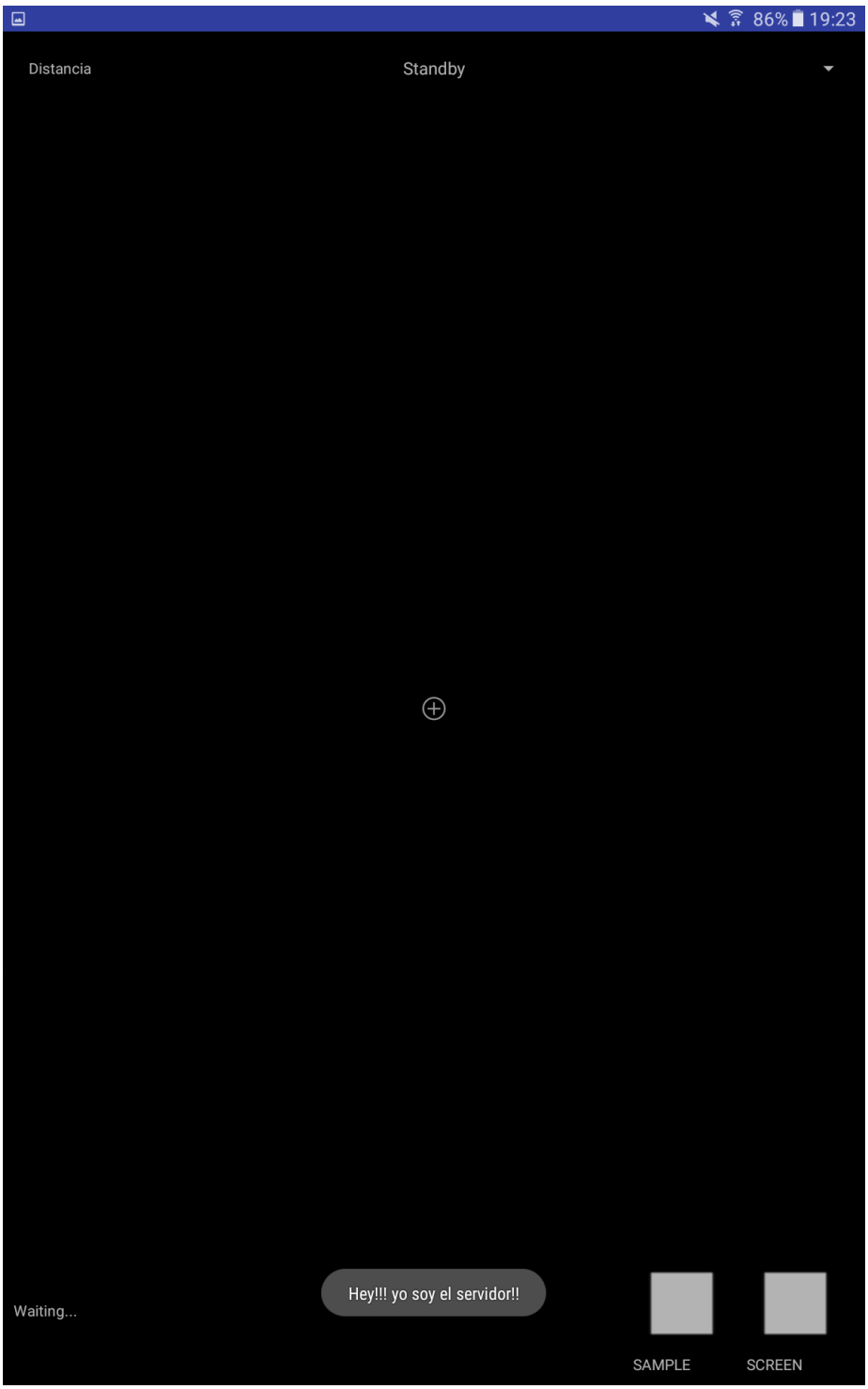

Figura B.1: Captura de pantalla del momento en el que se confirma la conexión entre dispositivos. En la parte inferior puede observarse un mensaje enviado por el dispositivo al que se acaba de conectar como confirmación.

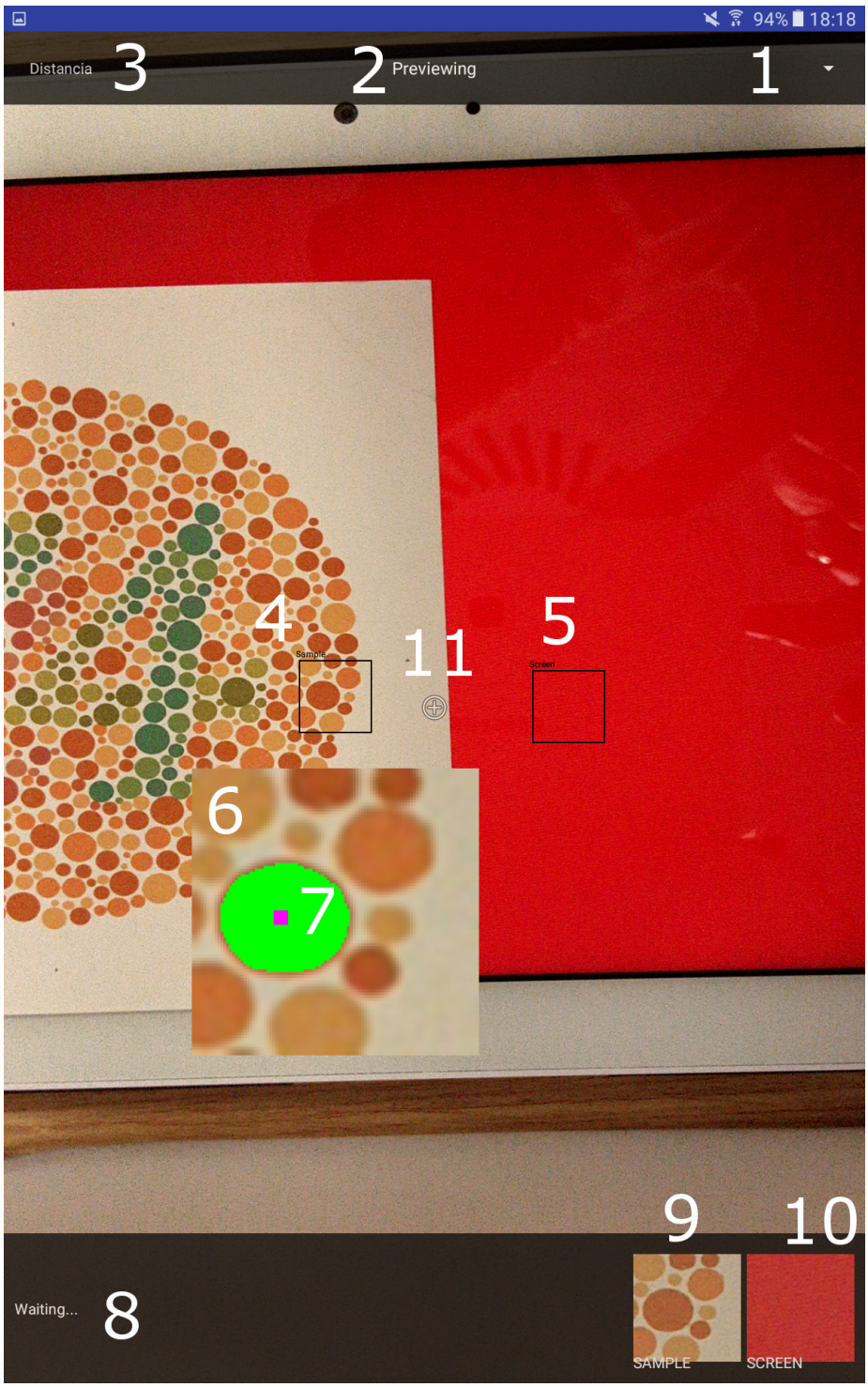

Figura B.2: Captura de pantalla de la interfaz anotada con los elementos numerados.

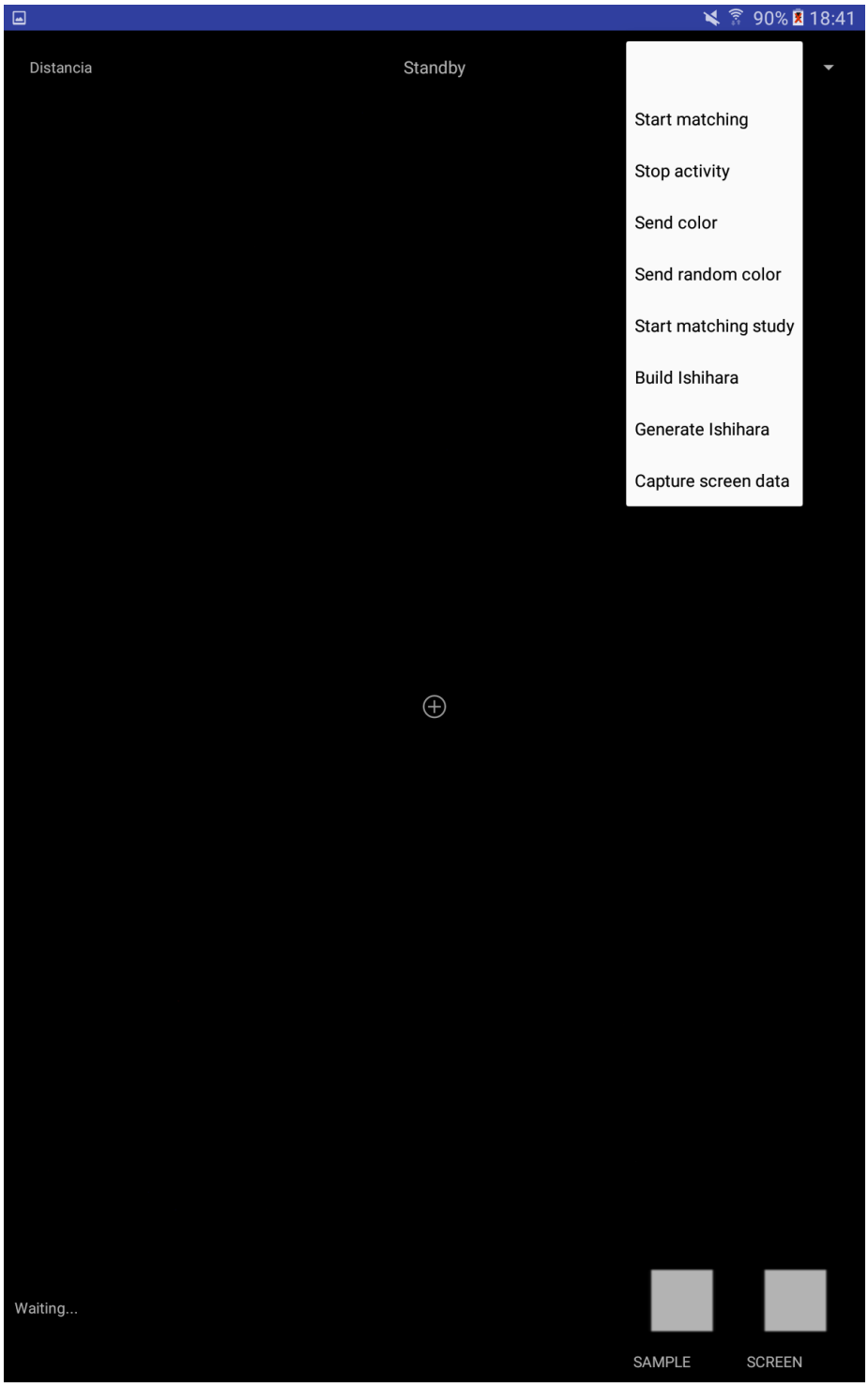

Figura B.3: Captura de pantalla del menú desplegado con sus opciones.

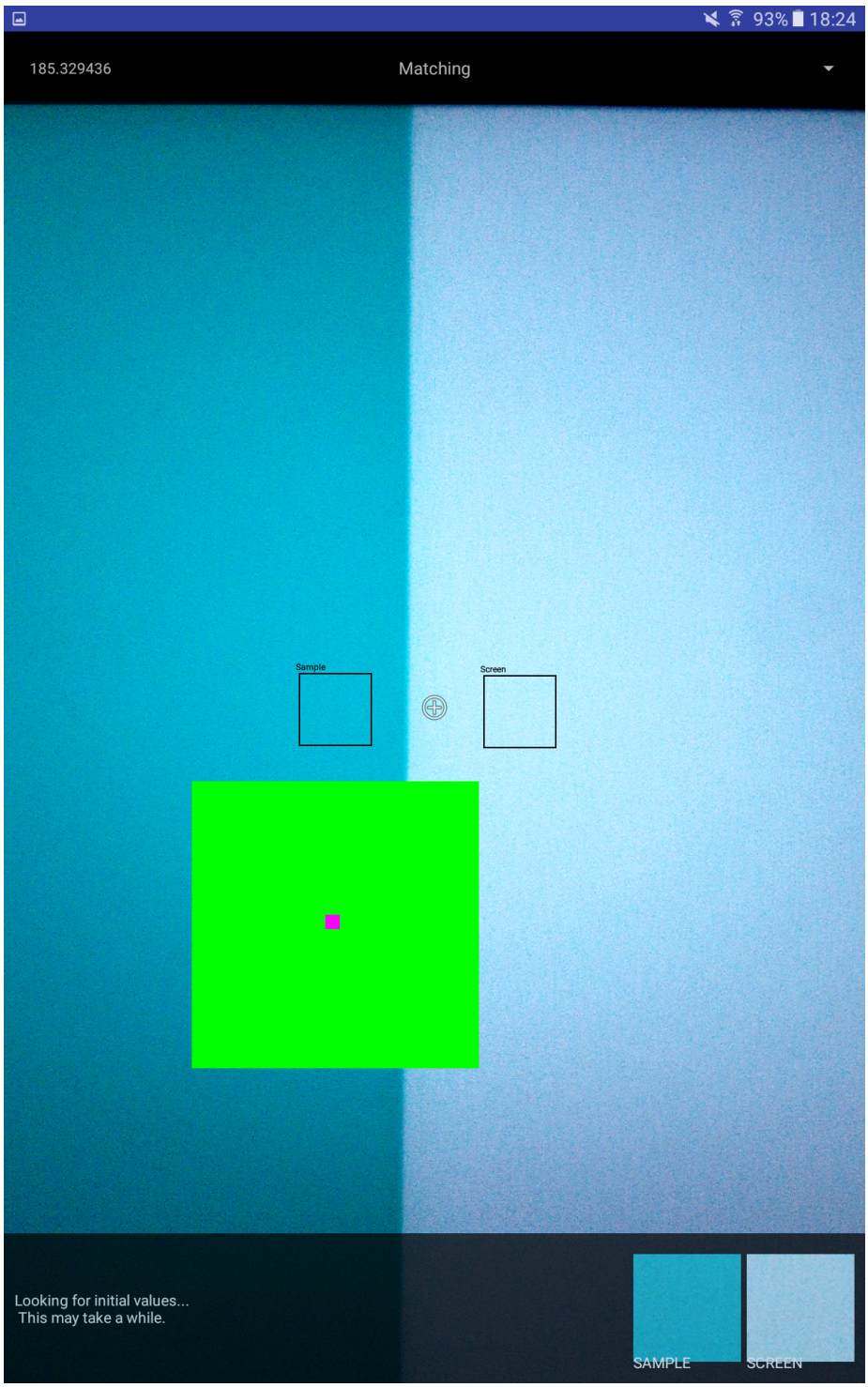

Figura B.4: Captura de pantalla en mitad de un proceso de igualación de color donde se observa la información actualizada de distancia, estado de algoritmo y de la aplicación, así como los colores y las cajas de recorte.

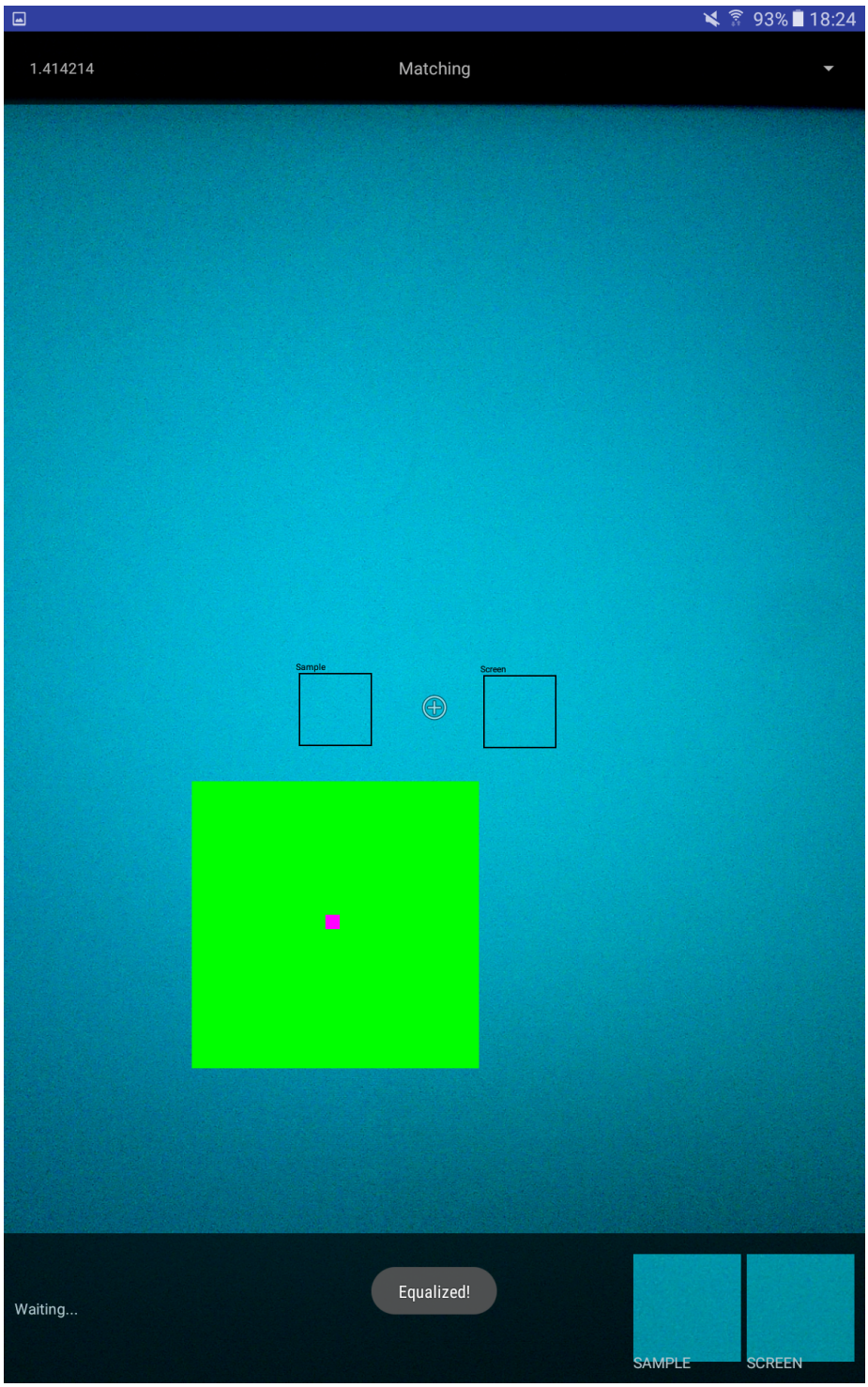

Figura B.5: Captura de pantalla del final del proceso de igualación de color, donde se observa el mensaje en la parte inferior de la pantalla.

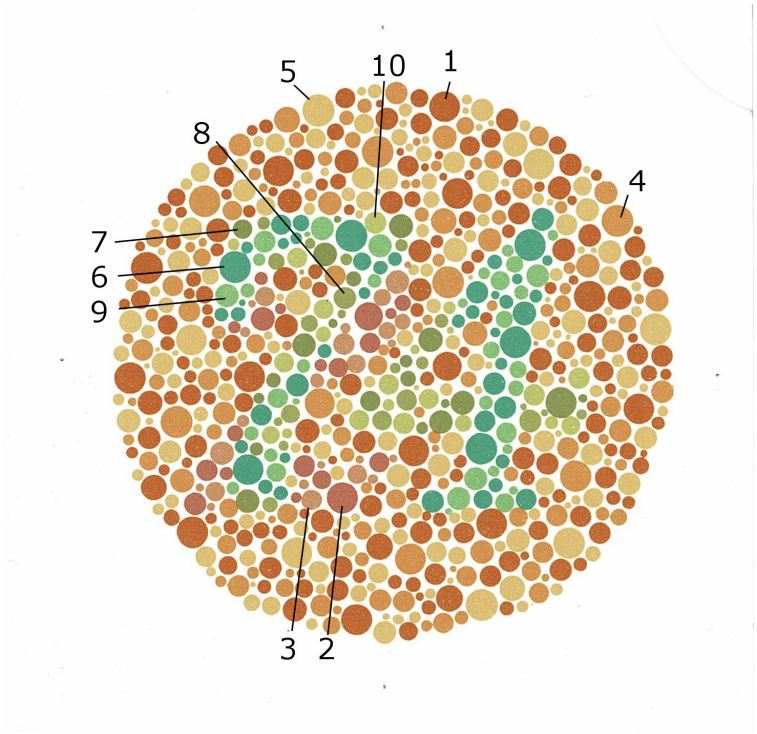

Figura B.6: Orden de los colores que se debe seguir en el proceso de creación de la lámina mostrada.

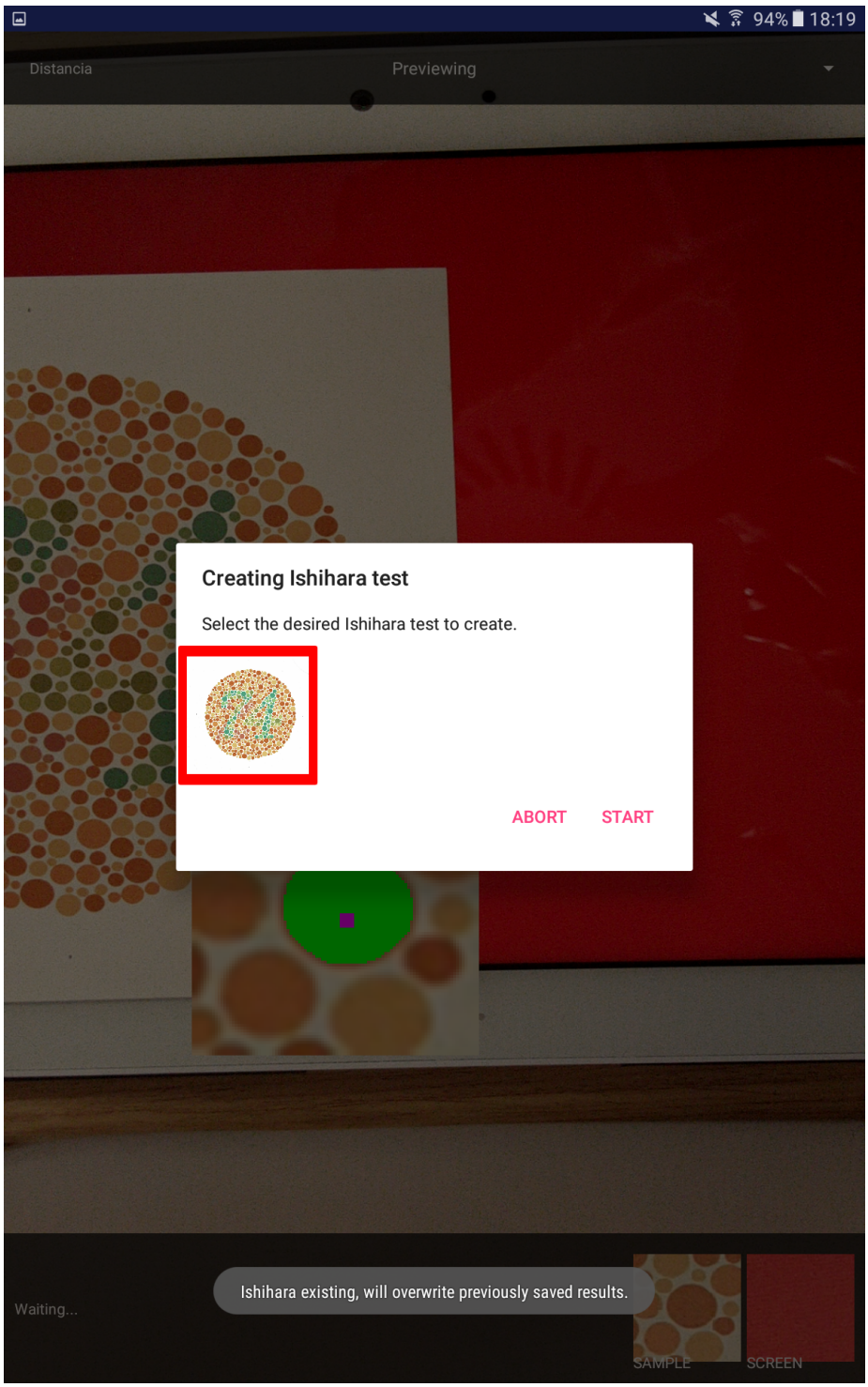

Figura B.7: Captura de pantalla de la selección de la lámina del test de Ishihara que se desea construir.

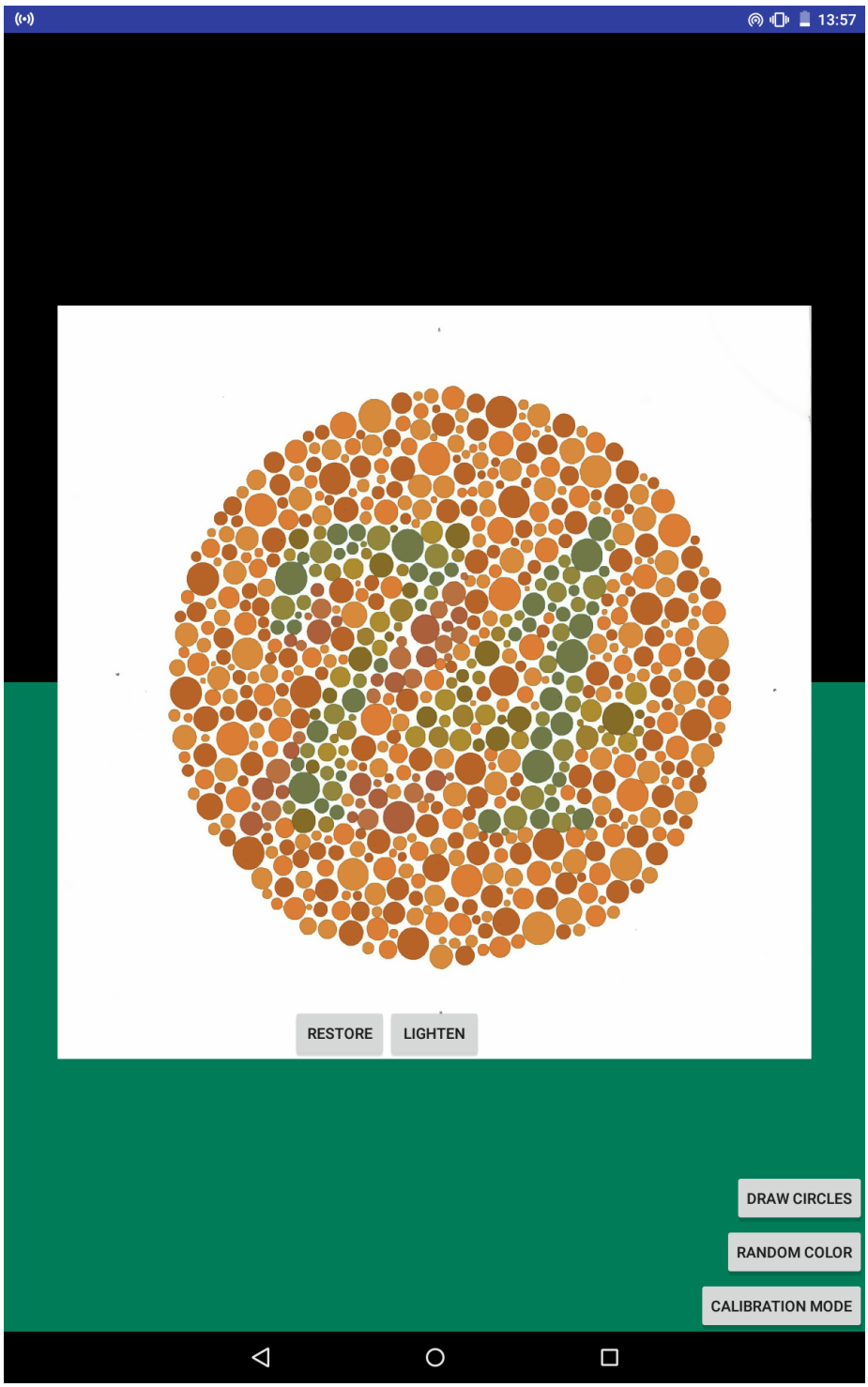

Figura B.8: Captura de pantalla del dispositivo *Pantalla* donde se muestra la lámina generada.

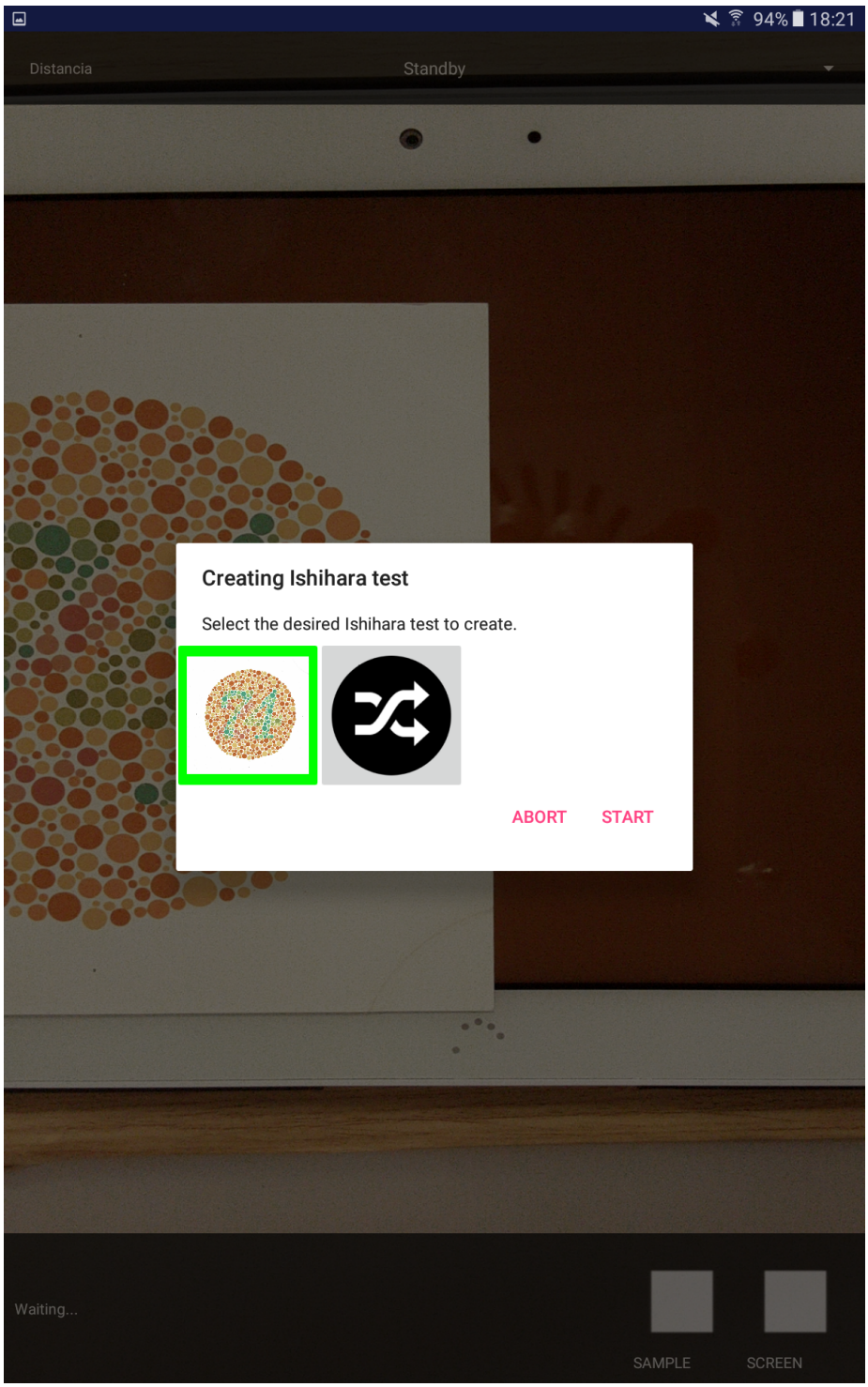

Figura B.9: Captura de pantalla del menú de selección de la lámina de Ishihara a generar.

Bibliografía

[1] OpenStax CNX. *Physics.* OpenStax College, 2018.

[2] Gunter Buxbaum. *Industrial inorganic pigments.* John Wiley & Sons, 2008.

[3] Bernice E Rogowitz, Thrasyvoulos N Pappas, and Jan P Allebach. Human vision and electronic imaging. *Journal of Electronic Imaging*, 10(1):10–20, 2001.

[4] Lindsay T. Sharpe, Andrew Stockman, Herbert Jaegle, and Jeremy Nathans. *Opsin genes, cone photopigments, color vision, and color blindness.* University Press, 2000.

[5] Joel Pokorny and Jennifer Birch. *Congenital and acquired color vision defects.* Grune and Stratton, 1979.

[6] Jennifer Birch. *Diagnosis of defective colour vision.* Butterworth-Heinemann, 2001.

[7] Germund Dahlquist. *Numerical Methods.* Dover Publications, 2012.